EXTRUDER DYNAMICS AND CONTROL IN LARGE SCALE ADDITIVE MANUFACTURING

A Dissertation
Presented to
The Academic Faculty

by

Chelsea Silberglied

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
George W. Woodruff School of Mechanical Engineering

Georgia Institute of Technology
December 2018

EXTRUDER DYNAMICS AND CONTROLS IN LARGE SCALE ADDITIVE MANUFACTURING

Approved by:

Dr. Thomas Kurfess, Advior
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Christopher Saldana
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Lonnie Love
Manufacturing Systems Research Group
Oak Ridge National Lab

Dr. Craig Blue
Energy Efficiency and Renewable Energy Programs
Oak Ridge National Lab

Date Approved: [November 20, 2018]

[To my friends and family]

ACKNOWLEDGEMENTS

I would especially like to thank my mother and father.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

AM	Additive Manufacturing
BAAM	Big Area Additive Manufacturing
BCS	Bead Characterization System
BG	Blue Gantry
CF-ABS	Carbon Fiber ABS
FDM	Fused Deposition Modeling
GUI	Graphical User Interface
LDV	Laser Dopler Velocitimeter
LSAM	Large Scale Additive Manufacturing
MAD	Mean Absolute Difference
MAE	Mean Absolute Error
MDF	Manufacturing Demonstration Facility
NRMSE	Normalized Root Mean Square Error
ORNL	Oak Ridge National Laboratory
PPR	Pulses Per Revolution
PRBS	Pseudo Random Binary Sequence
PWM	Pulse Width Modulation
RMSE	Root Mean Square Error
SIGUI	System Identification GUI

SUMMARY

Large Scale Additive Manufacturing has proven to be a good method for making near net shape parts. However, there are many part defects, such as bulging at corners and improper part seams that arise due to a lack of information about the extruder. In order to reduce the number of part defects, it will be necessary to both understand and control the extruder dynamics. To model the extrusion dynamics, a bead characterization system (BCS) was created. The BCS enabled measurement of the flow rate out of the nozzle. Tests were run to excite the dynamics of the extruder to perform system identification on various parts of the system. A second order underdamped system with two poles and two zeros was used to describe the relationship between the extruder RPM and flow rate out of the nozzle. While there is currently no physics based model of this response, the shear field developing, rapid changes in pressure with changing extrusion rates, and dynamics that occur when the bead hits the surface can be used to explain this second order response. This model had an over 80% normalized root mean squared error when compared to validation data. After the model was created, a feed forward controller was implemented which used the created models to predict the flow rate and command part of the system to maintain a constant bead width. This controller proved to control the bead geometry while in the middle of an extrusion pattern, producing beads with six times less variation when compared to experiments run without a controller. Future work will need to be completed to study the dynamics of the starting and stopping conditions for the extruder, but the BCS has proven to be a viable method of collecting data about the extrusion rate to bead geometry controls.

CHAPTER 1. BACKGROUND AND INTRODUCTION

This chapter presents a review of additive manufacturing, system identification, and controls. Fundamentals from these areas were used to create a system model and controller for Large Scale Additive Manufacturing extruders, which is presented later in the thesis. This chapter also discusses some previous work that was performed to understand extruder dynamics.

1.1 Additive Manufacturing

Additive manufacturing (AM) can refer to any manufacturing process that adds material instead of removing material. There are seven main types of additive manufacturing that add material in a layer by layer process. These seven types include binder jetting, directed energy deposition, material jetting, vat photopolymerization, powder bed fusion, sheet lamination, and material extrusion. [1] This thesis will focus on material extrusion as a form of additive manufacturing. Additive manufacturing is commonly used for rapid prototyping because it offers a relatively low-cost solution to obtain a tangible object. [2]

Fused deposition modeling (FDM) is a process in which heated thermoplastic filament is extruded to trace a pattern one layer at a time. When one layer is finished printing, the print head moves up in the z direction to deposit the next layer. This is a very similar process to the Big Area Additive Manufacturing (BAAM) discussed in this thesis.

While additive manufacturing can be used to rapidly prototype a part, there are problems with part accuracy. The three main causes of error are mathematical, process, or material related. [2] Mathematical errors usually occur due to the way that a part is processed before

printing while process and material errors occur during the printing process. This thesis discusses and explores a way to model and control errors due to the process and materials used in BAAM.

1.2 Big Area Additive Manufacturing/Large Scale Additive Manufacturing

Large Scale Additive Manufacturing (LSAM) is a type of material extrusion which enables parts several meters in size to be printed. [3] LSAM utilizes a single-screw extruder which allows for an extrusion rate of 50 kg/hr with a 7.6mm nozzle. This deposition rate can be higher depending on the material. [4] This makes LSAM attractive as a manufacturing method over smaller scale AM. Another way in which LSAM will help to make AM more attractive as a manufacturing method is due to lower material cost when compared to smaller scale 3D printing technologies, in which material can cost $100-$200 USD/kg of material. On the other hand, the pellets for LSAM can cost between $2-11 USD/kg. [5], [6] So far, LSAM has been applied to making prototype electric vehicles and for mold production. [7], [8] LSAM allows for over a 50% time savings for low production or prototype molds, when compared to molds created using subtractive manufacturing. [9]

1.2.1 Printing Process

This section will discuss the process for material extrusion. The pellets of material, which can be seen in Figure 1a, are stored in gaylords, seen in Figure 1b. From the gaylords, the material is vacuumed into a drier. Before the material can be printed, it must be dried for at least 4 hours to eliminate moisture from the pellets. The material driers can be seen in Figure 1c. If the pellets are wet when extruded the material will be fuzzy and will not have

good properties. After the material is dried, the print head, seen in Figure 1d will call for more pellets to be sent once the pellets in the hopper fall below a certain threshold.

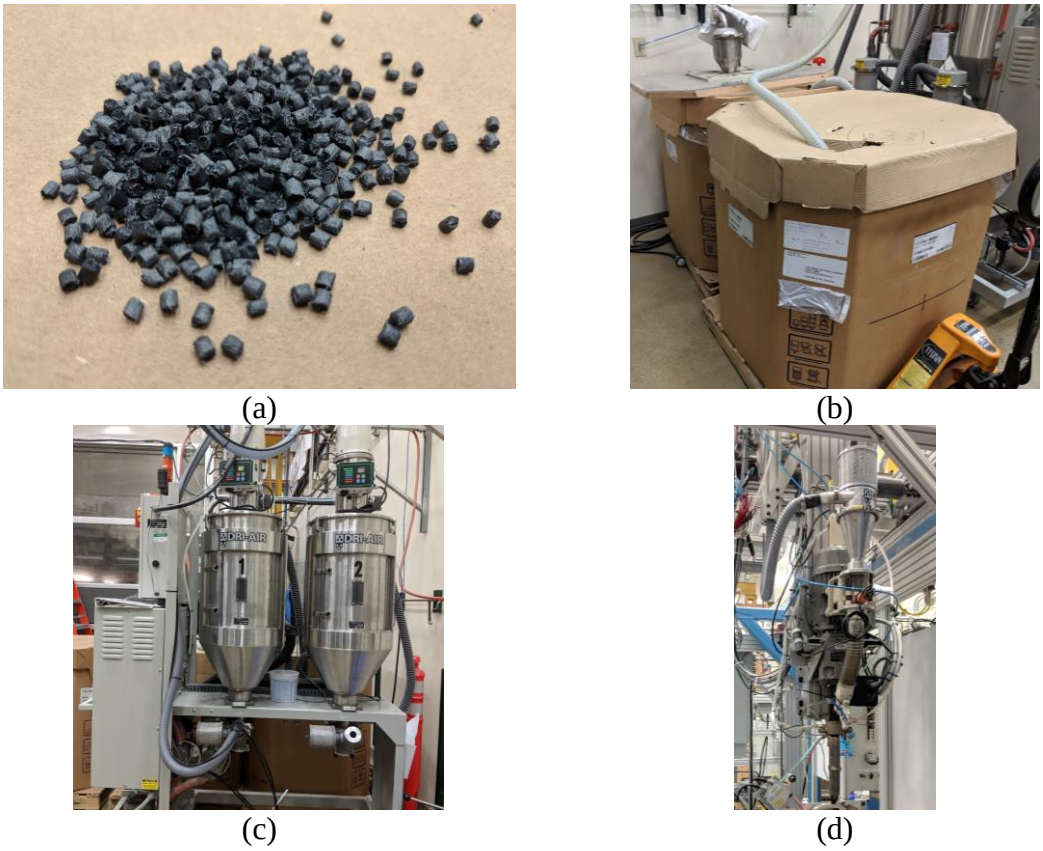


Figure 1. Material Flow in Printing Process

1.2.2 Materials Used in BAAM

BAAM uses different polymers as the printing material. There are certain properties that a material must have in order to produce successful prints. The material must be able to be pushed out of the nozzle at a desired flow rate without exceeding the pressure limit of the system, the material must be able to maintain a free standing bead, and the material must be able to support layers printed above it. [4] It can be difficult to predict the behavior of a polymer due to its viscoelastic properties. In a molten state, polymers are mostly viscous, where energy used in deformation is immediately dissipated, and partially elastic, which

holds onto some of the energy used in deformation. Many materials have been tested with BAAM systems, but so far it seems that carbon fiber reinforced ABS (CF-ABS) has good properties for applications such as low temperature tooling. [10] The materials used in BAAM are normally non-Newtonian which means that viscosity is dependent on the shear rate. This can make it difficult to predict the way the material will behave.

1.2.2.1 Extrudate/Die Swell

Die swell is a common occurrence in polymer melt extrusion. The viscoelastic property of a polymer is mostly responsible for this phenomenon. Elastic recovery is the main cause of die swell, in which the polymer expands once exiting the die. Figure 2 shows a representation of die swell. It can be seen in the figure that another cause of swell is a change of the velocity profile from a parabolic shape to a straight velocity profile once the material exits the die. The amount of die swell is highly dependent on the material properties. For example, PVC exhibits only about 10-20% die swell while other materials can swell 300%. Currently, it is very difficult to devise a mathematical model of die swell that is applicable in engineering applications. [11]

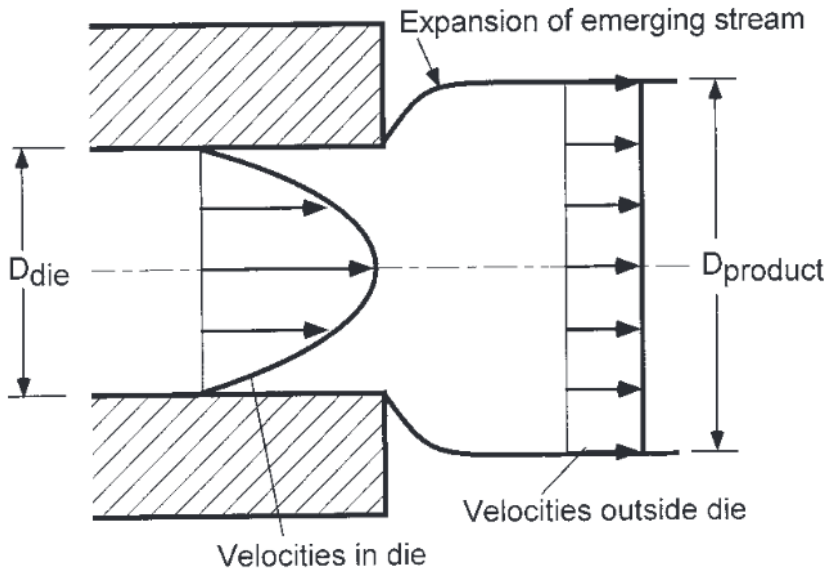


Figure 2. Die Swell Phenomena [11]

1.2.3 Extruders

Extruders are responsible for about 60% of the world's plastics to create materials such as pipes, tubes, and sheets. [12] The purpose of an extruder is to transform solid feedstock into a homogeneous melt and push it through a die at a constant rate. Extruders can broadly be defined as continuous or discontinuous in their mode of operation. For the purposes of this paper, discontinuous extruders will not be discussed because they are not currently used in LSAM. The two main types of extruders used in LSAM are single screw extruders and gear pump extruders. Each type offers different benefits for the printing process. Other types of continuous extruders will be discussed as well.

1.2.3.1 Diehead Pressure

Die head pressure is the pressure required to force the material through the die at the end of the extruder. This required pressure is independent of the type of the extruder used. An extruder must be capable of generating enough pressure to force the material though the die. The diehead pressure is determined by the shape of the die, temperature of the polymer, flow rate, and the rheological properties of the polymer.

1.2.3.2 Single Screw Extruders

A single screw extruder is made up of a single rotating screw which is situated in a heated barrel. This causes frictional forces on the material, screw and barrel which move the material through the extruder and provide most of the heat for the process in the form of frictional and viscous drag. The solid feedstock takes up more volume than the liquid extrudate due to inefficient packing, and a constant mass flow rate must be maintained throughout the screw. Therefore, the screw channel area decreases as the material moves through the screw. The ratio of the first channel depth to the smallest channel depth is known as the compression ratio which is usually between 2 and 4. [11], [12]

The screw is comprised of three sections, a feed, compression, and metering section. Each section of the extruder is responsible for a different part of the process. A representation of the screw can be seen in Figure 3. As the material moves throughout the screw, it will go through different functional zones which are different than the zones of the extruder. The zones of the extruder are due to the design of the screw, and the zones of the material are due to the material properties. The three functional zones for the material are the solids conveying zone, the plasticating zone, and the melt conveying zone. The screw is designed to have each of the material's functional zones take place in a different region of the screw,

but the actual location of the change may lie somewhere between the physical change in screw geometry.

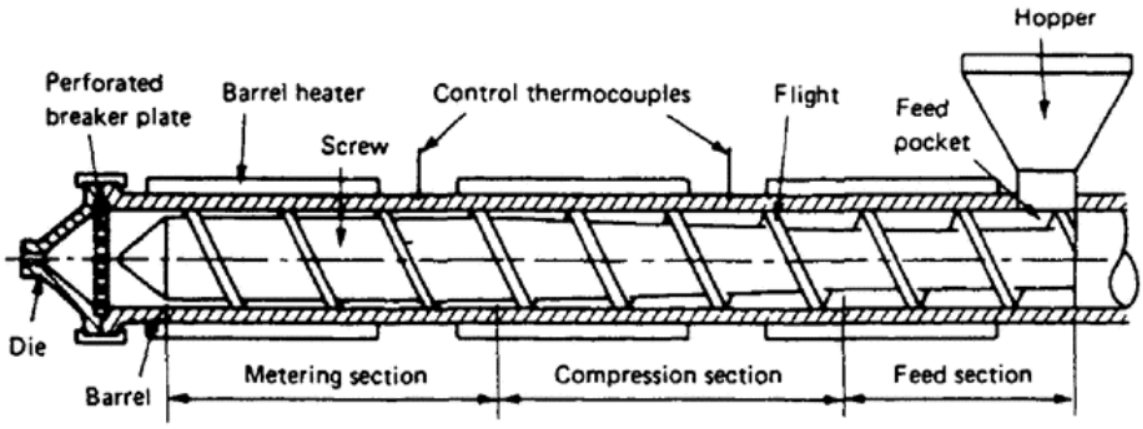


Figure 3. Single Screw Extruder [12]

To move material through the barrel, material is usually gravity fed into the extruder barrel from a hopper. The feed section is where the material enters the extruder, and typically is where the solids conveying zone lies. The plasticating zone starts somewhere between the feed and compression zone. The exact location of this transition depends on the polymer properties, machine geometry and operating conditions. Finally, once all of the solid pellets have melted, the melt conveying zone begins near the metering section of the screw. Once the material has reached the end of the metering section, it is pushed through die if sufficient diehead pressure is reached. [11]

1.2.3.3 Twin Screw Extruders

There are many different types of twin screw extruders. Twin screw extruders have two Archimedean screws which can either mesh or not mesh in order to extrude material. [11] As previously stated, the main method of material transport through a single screw extruder

is frictional and viscous drag, which makes the flow through the extruder very polymer dependent. Twin screw extruders rely on other methods of transport which make them better for materials without favorable frictional properties. On the other hand, intermeshing twin screw extruders are a partially positive displacement process. A problem with twin screw extruders is that it is difficult to analyze the flow within the extruder. Therefore, many times twin screw extruders will come with modular screws which can be interchanged to achieve the desired properties.

1.2.3.4 Vented Extruders

As material is moved through the extruder, volatiles can build up. A vented extruder allows for volatiles to escape from the extruder which can help to eliminate air pockets in the extruded material. Twin screw extruders can vent solvent contents of over 50% while a multiport single screw extruder can only handle about 5% solvent contents. [11] As such, it can be difficult to properly vent a single screw extruder without multiple ports. Vents can also be used to add components, such as fillers, additives, and reactive components to the polymer in the extruder.

1.2.3.5 Gear Pump Extruders

Gear pumps are used in conjunction with either single or twin screw extruders. At times, they are referred to as positive displacement extruders, but this is not strictly true because there are clearances between the gears and the housing. Gear pump extruders are generally less pressure sensitive to pressure fluctuations than a single screw extruder. However, gear pumps can make the output from the screw much more reliable and are often added onto machines where output fluctuations are over $\pm$1%. [11] The LSAM printer from

THERMWOOD, seen in Figure 4, utilizes a gear pump extruder, which allows for a more consistent bead geometry and control over the flow rate. [13] A down side to gear pump extruders is that it can cause damage to the additives such as the fiber in CF-ABS in the printing process.

Figure 4. THERMWOOD LSAM Printer with Gear Pump Extruder [14], [15]

1.3 Part Defects in BAAM

There are many different types of defects in BAAM that degrade part quality. These defects take place at different points during the printing process, namely there are defects at the beginning, middle, and end of an extrusion. This thesis focuses on addressing the defects caused in the middle of the printing process. Some methods of dealing with these defects will be discussed.

1.3.1 Defects from the Beginning of an Extrusion

There are two main types of defects that occur with the beginning of extrusion. The time delay for material to exit the nozzle at the beginning of an extrusion changes based on the residence time in the extruder. Therefore, as the gantry moves about the part, there is always a different amount of time between extrusions depending on how far the print head needs to travel to get to the next point. This can cause a void where material is not printed where it is expected to be, and if the gantry moves slowly, it can cause a bulge at the beginning of an extrusion. Figure 5 shows a void in the part in the red box due to a lack of information about when material will be extruded from the print head relative to when it was commanded to print.

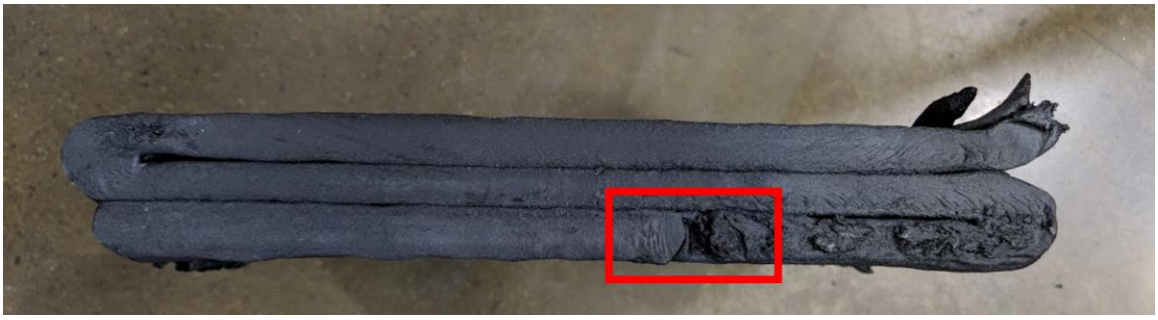


Figure 5. Void in Part Due to Unknown Time Delay

1.3.2 Defects in the Middle of Extrusion

There are different part defects caused in the middle of an extrusion. Most of these defects occur when traversing a corner. When the gantry moves around a corner, it slows down on the entrance to the corner and then speeds up as it exits the corner. Currently, when traversing a curve there is no command to change the extruder output, so the bead is larger when entering the corner and narrow when exiting the turn. When there is a command to change the extruder output, it is not properly timed because the response of the extruder is

unknown. The research presented in this thesis has to do with changing the feed-rate in coordination with the extruder command to maintain a constant bead width. Figure 6 shows a part which is supposed to have a constant height. It can be seen that at the corner there is a protrusion which is pointed out in red.

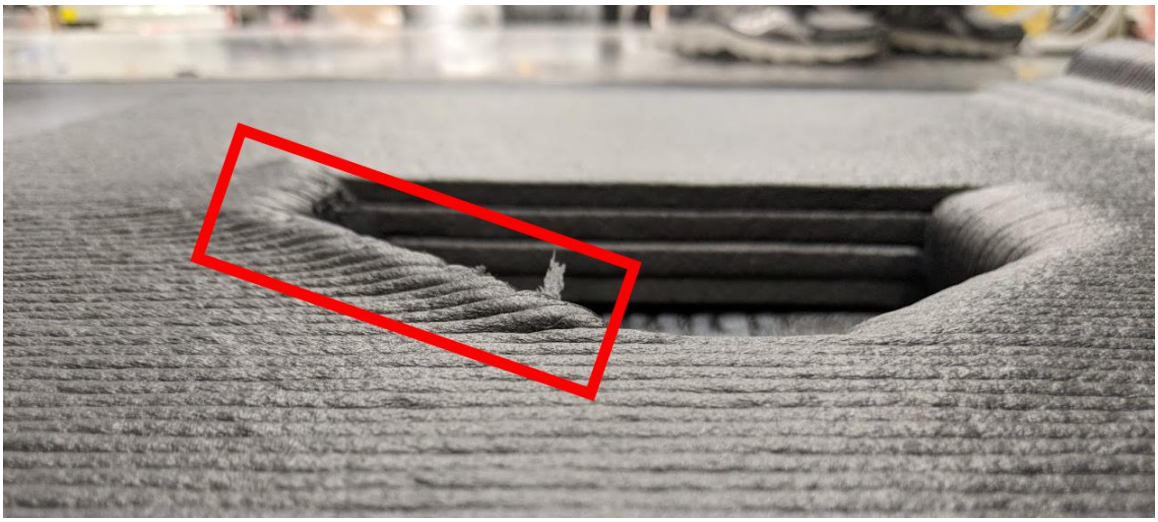


Figure 6. Bulge Due to Changing Speed at Corner

1.3.3 Defects at the End of an Extrusion

At the end of an extrusion, material continues to come out of the nozzle even though the screw is not turning anymore. For the purposes of this thesis, this phenomena will be known as “drool.” This causes multiple types of part defects which can be seen in Figure 7. On the left, over-extrusion can be seen where material is still coming out of the extruder as the print head moves away from the part. On the right, over extrusion causes a part defect in which material is extruded and the extruder stays in the same position. When the extruder lifts away from the part, it brings some of the material up with it.

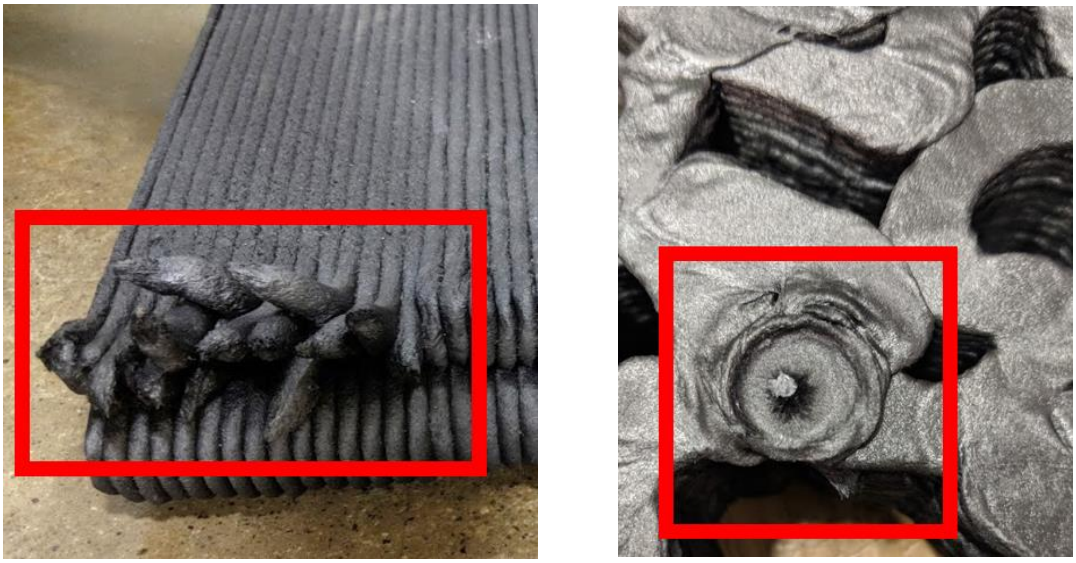


Figure 7. Part Defects Due to Drool

Many times the start and end of an extrusion will occur next to each other in the part geometry. This causes defects along the seam, similar to that seen in Figure 8. Here, the layers should be smooth and a uniform thickness. Along the region boxed in red, the extruder stopped and started to extrude material. For each of the layers, the extruder was commanded to start and stop at the same time, but it is clear that the defects are causing the material to take a different form on each subsequent layer.

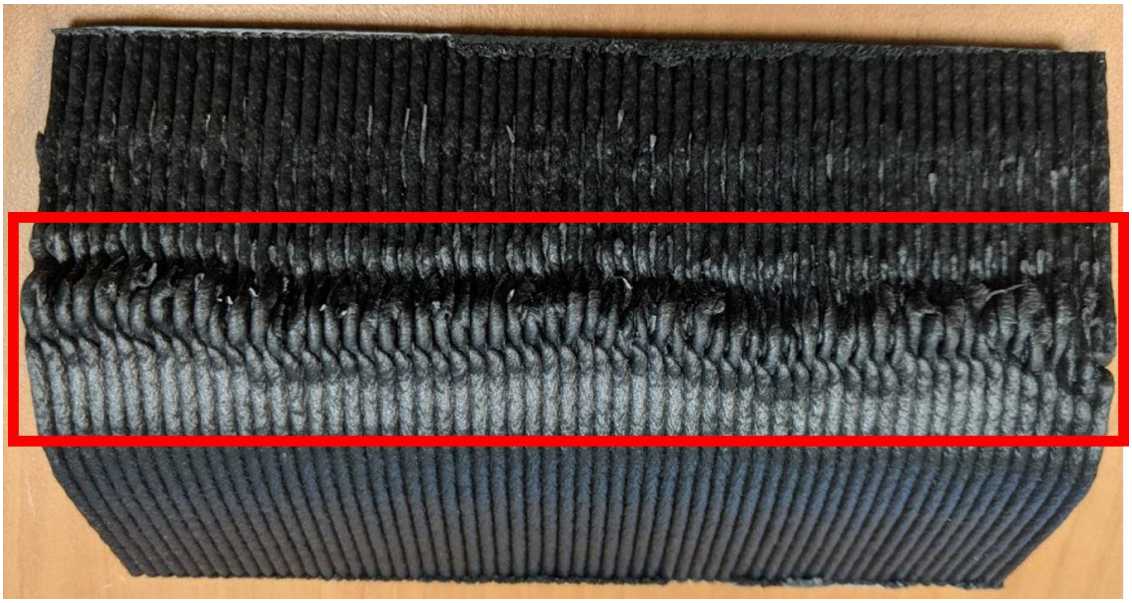


Figure 8. Defect along Part Seam where Extrusion Starts and Ends

1.3.4 Current Methods for Reducing Amount of Part Defects

While there are currently problems with the extrusion process, steps have been taken to reduce the number of part defects. For example, to minimize the effect of the delay when starting to extrude, a dwell time is input into the controller. When the starting dwell time is enabled, the extruder is commanded to extrude before the gantry moves along the printing path. This reduces the amount of voids during start-up, but it does not account for the time variant aspect of beginning to extrude. Therefore, this can lead to bulging if the residence time is low, or a void if the residence time is high.

A similar method of an ending dwell time is used when the extruder is about to stop extruding. When the ending dwell time is enabled, the extruder will stop extruding the prescribed amount of time before it gets to the end of a toolpath so that the drool out of the nozzle will fill in the toolpath. This tends to work well, but the amount of drool is different

for each extruder nozzle size so this is not a perfect approach. Both the starting and ending dwell time are user defined parameters that can be changed in the middle of a print. Therefore, an operator will dial these values in for each print until they see the desired behaviour from the extruder. This requires experienced operators who know what to look for in the printing process.

A third method of dealing with part defects is a mechanism known as the tamper. Figure 9 shows the tamper attached to the extruder. The tamper fetures a motor connected to a platen that moves up and down to hit the printed material down.[16] The bottom of the platen's stroke is usually aligned with the nozzle so that as the print head moves about the part, the tamper knocks down any defects that could interfere with the nozzle. Air is delivered to the tamper through the pneumatic hose. The air moves through internal channels in the tamper and exits through air outlets. This works to keep the tamper cool to prevent material from sticking to the mechanism during a print.

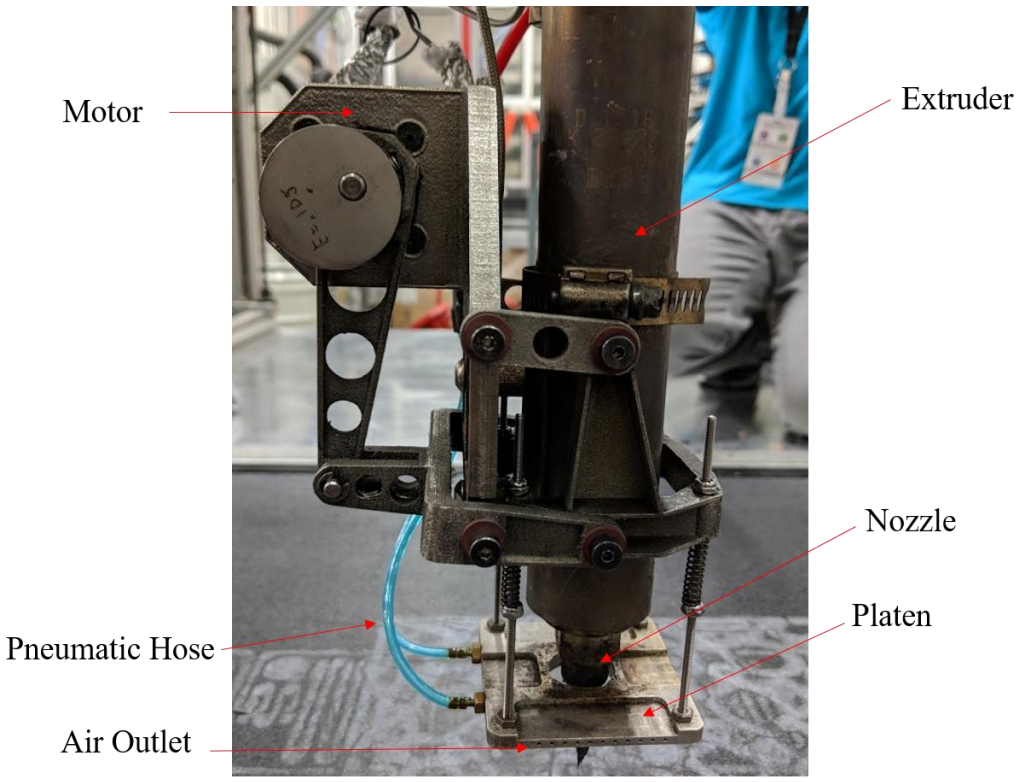


Figure 9. Tamper Mechanism to Level the Print

1.4 Noise and Filtering

When data is collected, there will almost always be associated noise. It is important to eliminate as much noise as possible in the experiment setup. For example, the system should be checked to make sure that there are no grounding issues and shielding should be used on cables. [17] Inevitably, there will be noise and system disturbances that are unexpected. For example switching signals such as Pulse Width Modulation (PWM) and Space Vector Modulation, which are commonly used by servo drives, generate electrical noise due to high switching frequencies.

The goal of filtering is to remove as much noise as possible from the data without changing the meaning of the data collected. [18] This can be done through both hardware, such as

ferrite cores, and software. There are many basic filters which can be applied to reduce the amount of noise in the data as well as more sophisticated filters that can be applied.

1.4.1 Outliers and the Hampel Filter

Outliers can impact analysis so it is important to recognize their existence and deal with them appropriately. [19] There are two main ways of handling outliers. The first method is to recognize outliers and replace them with more reasonable values. This allows for data processing techniques that are do not require data without outliers to be used. Another option for dealing with outliers is to perform analysis procedures that are not sensitive to outliers. The Hampel filter works to replace outliers with sensible values.

Moving median filters have proven to be extremely useful in many applications despite only having one tuning parameter. [20] The Hampel filter is closely related to a median filter. Both of these filters use a moving data window which can be represented by the following equation:

$$W_k^K = \{x_{k-K}, \dots, x_k, \dots, x_{k+K}\} \tag{1}$$

where K is the window half width and takes the value of a positive integer. For any moving window, the median value can be defined as m_k, seen in Equation 2.

$$m_k = median\{x_{k-K}, \dots, x_k, \dots, x_{k+K}\} \tag{2}$$

The Hampel filter response can be given by

$$y_k = \begin{cases} x_k, & |x_k - m_k| \leq tS_k \\ m_k, & |x_k - m_k| > tS_k \end{cases} \quad (3)$$

where t is a scalar threshold, S_k is the mean absolute difference (MAD) scale estimate, which can be defined according to the following equation. [20], [21]

$$S_k = 1.4826 \times median_{j \in [-K,K]}\{|x_{k-j} - m_k|\} \quad (4)$$

The value of 1.4826 makes the MAD estimate equal to the standard deviation for normally distributed data. The value of t in Equation 3 makes the filter more or less aggressive. A more aggressive filter will remove more outliers, but also may change . In addition, it is necessary to pick an appropriate window length for the Hampel filter. If a window length is too short, then outliers may not be detected. Another problem that could occur if the window length is too short is that an outlier could be replaced by an incorrect median value which is influenced by data within the window. [21]

1.4.2 Savitsky-Golay Filtering

The Savitsky-Golay filter fits a polynomial over a number of samples using a least squares approximation. [18], [22]–[24] This filtering method creates a low pass filter which is generated from Equations 5 and 6. Figure 10 shows a series of samples, $x(n)$ from a signal as solid dots. To obtain the polynomial of the dotted line shown in the figure, a sample set of $2M + 1$ samples were selected that are centred around $n = 0$.

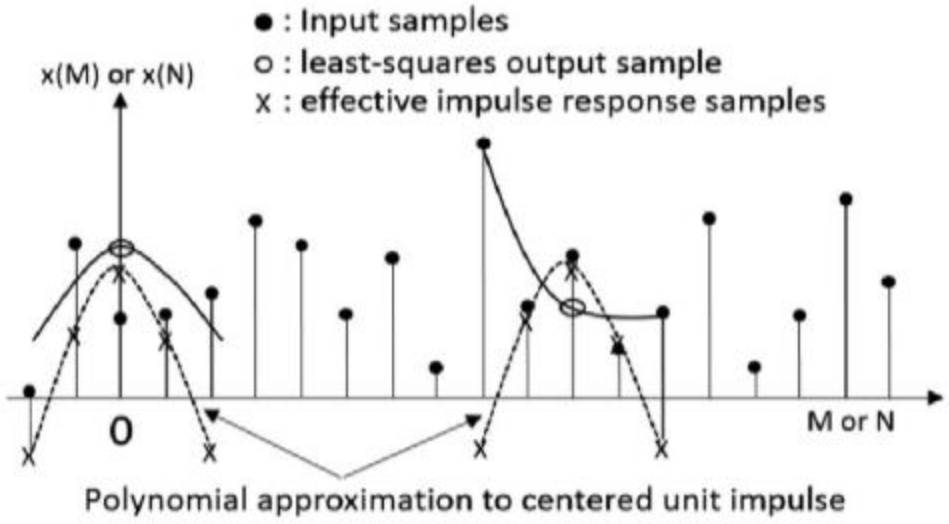


Figure 10. Savitzky-Golay Filter Representation [23]

The coefficients of this polynomial can be found according to Equation 5. [24]

$$p(n) = \sum_{k=0}^{N} a_k n^k \tag{5}$$

The mean squared approximation error,ϵ_N, can be minimized by the Equation 6

$$\epsilon_N = \sum_{n=-M}^{M} (p(n) - x[n])^2 = \sum_{n=-M}^{M} \left(\sum_{k=0}^{N} a_k n^k - x[n] \right)^2 \tag{6}$$

It is not necessary that the interval is centered around $n = 0$ when applying the filter. The process Savizky-Golay filtering was summarized in "On Savitzky-Golay Filtering for Online Condition Monitoring on Transformer On-Load Tap Changer" as the following three steps: [23]

1. Right shift the interval by one sample

2. Set a new origin as the middle position of $2M + 1$ samples
3. At the new origin, perform polynomial fitting and evaluation for every sample in the sample set.

The Savitzky-Golay filter is powerful because it can preserve the waveform of an oversampled signal that is corrupted by noise. This filter does not introduce any feature shift with respect to the original signal because it is symmetric and has zero phase. MATLAB has a Savitzky-Golay function, sgolayfilt(), which can be used to implement symmetric and non-symmetric Savitzky-Golay filters, which was utilized in the data processing for this work. [22], [24]

1.5 System Identification and Controls

A large portion of the work performed relied on system identification in order to come up with a model to describe the system. This section discusses some of the theory that was used in order to perform calculations and create a model.

1.5.1 Transfer Functions

Transfer functions are used to describe the relationship between a system inputs and system outputs without solving a differential equation. They are important for determining dynamic system responses. [25] A transfer function can be defined in many different ways. Equation 7 shows a representation of a transfer function $G(s)$, where s is a complex variable represented by $s = \sigma + j\omega$.

$$G(s) = \frac{b_m s^m + b_{m-1} s^{m-1} + \cdots + b_1 s + b_0}{a_n s^n + a_{n-1} s^{n-1} + \cdots + a_1 s + a_0} e^{-s\tau} \quad (7)$$

The coefficients, a_i and b_i, in the previous equation are real numbers and τ is the time delay in seconds. It is often useful to define a transfer function in terms of its poles and zeros. Therefore, it is common to represent Equation 7 in the form shown in Equation 8, otherwise known as the zero pole form. [26]

$$G(s) = \frac{N(s)}{D(s)} = K \frac{(s - z_1)(s - z_2) \ldots (s - z_{m-1})(s - z_m)}{(s - p_1)(s - p_2) \ldots (s - p_{n-1})(s - p_n)} \quad (8)$$

In the previous equation, $N(s)$ and $D(s)$ are the numerator and denominator respectively, z_i represent the system zeros, and p_i represent the system poles, K is the gain constant which is defined by the following equation.

$$K = \frac{b_m}{a_n} \quad (9)$$

The zeros and poles of an equation are found by solving $N(s) = 0$, and $D(s) = 0$ respectively. Once the poles, zeros, and gain constant of a system are known, it is possible to completely characterize a system. [25] Poles and zeros usually are real numbers or appear in complex conjugate pairs.

1.5.1.1 Homogeneous Response

A homogeneous response of a system is the response that occurs when a system is given a set of initial conditions and then has no other inputs. [27] This response can be described by a series of weighted sums, seen in the following equation:

$$y_h(t) = \sum_{i=1}^{n} C_i e^{\lambda_i t} = \sum_{i=1}^{n} C_i e^{p_i t} \tag{10}$$

where y_h is the homogeneous response, C_i is determined from the set of initial conditions, t is time, and λ_i are the roots of the characteristic equation and p_i are the system poles. The characteristic equation can be seen in Equation 11.

$$D(s) = s^n + a_{(n-1)}s^{n-1} + \cdots + a_0 = 0 \tag{11}$$

It can be seen that the roots of the characteristic equation are the same as the poles for the system. This means that $\lambda_i = p_i$, which can be seen in the exponents of Equation 10. Knowledge of the system poles is very powerful because the pole locations give the natural response of the system.

Figure 11 shows a poles-zero plot with a real x-axis and an imaginary y-axis. The poles are shown as crosses (×) on the plot. The pole locations can lead to a stable, marginally stable, or unstable system response. The values that the poles take on to create either a stable, marginally stable or unstable system response was characterized in Understanding Poles and Zeros, and can be seen below: [25], [28]

A. Stable Pole Values

1. Real Negative ($p_i = -\sigma$): This will lead to an exponentially decaying homogeneous response. If the pole is more negative, or further away from the origin, then the response will be faster while poles that are close to the origin will decay slower. The response can be described by $Ce^{-\sigma t}$ component in the homogeneous response.
2. Complex Conjugate Pair in the Left Half Plane ($p_i, p_{i+1} = -\sigma \pm j\omega$): This leads to a decaying sinusoid in its homogeneous response which can be represented by $Ae^{\sigma t}\sin(\omega t + \phi)$. For this response, decay rate will be specified by σ, the oscillation frequency will correspond to ω, and A and ϕ are determined by the initial conditions.

B. Marginally Stable

1. Zero ($p_i = 0$): A pole located at the origin will not change from its initial conditions. This response can be described by a $Ce^0 = C$ component in the homogeneous response.
2. Imaginary Pole Pair ($p_i, p_{i+1} = \pm j\omega$): It can be seen that an imaginary pole pair has a zero real component. Therefore, it lies on the imaginary axis, which creates a marginally stable response. An imaginary pole pair creates a component with the equation $Asin(\omega t + \phi)$ in the homogeneous response, which is an oscillatory response with constant amplitude.

C. Unstable

1. Real Positive ($p_i = \sigma$): This is a real pole in the right half plane. This leads to an unstable response in which the system will have an exponentially increasing component $Ce^{\sigma t}$ in its homogeneous response.

2. Complex Conjugate Pair in the Right Half Plane($p_i, p_{i+1} = \sigma \pm j\omega$): This leads to an exponentially increasing component in the homogeneous response. Similar to that presented in the complex conjugate pair in the left half plane, this response can be represented by the same equation, $Ae^{\sigma t}\sin(\omega t + \phi)$. The only difference is that now σ will describe the rate of exponential rise.

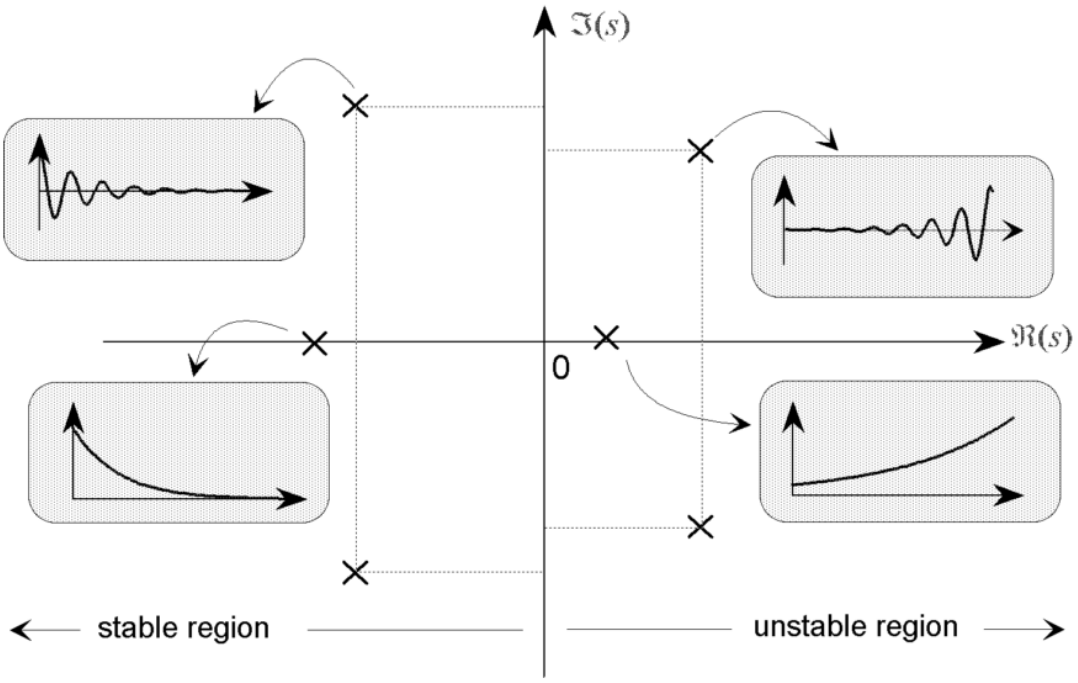


Figure 11. System response with different values for poles [25]

As seen above, stable poles occur in the left half plane, and have negative real components to the poles. Marginally stable systems have a zero real component in their poles, and unstable systems have positive real components. If a system is made up on multiple poles, it is characterized by its least stable pole. For example, if a system has two complex conjugate poles in the left half plane and a zero pole, the system will be considered marginally stable.

1.5.1.2 Second Order Systems and Damping

Second order systems can be seen in mechanical systems that have two energy storage elements, such as a mass spring damper system. In a mass spring damper system, such as that seen in Figure 12, energy is stored in both the spring and the damper. [29]

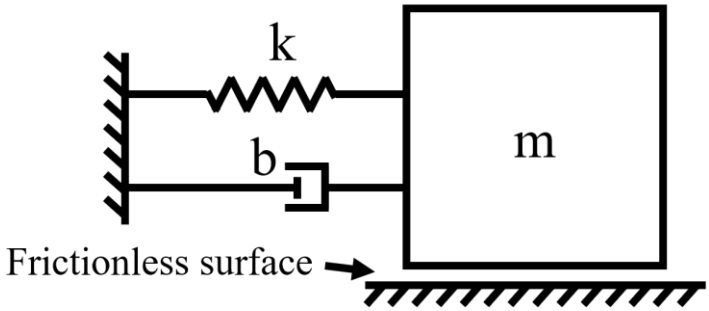


Figure 12. Mass spring damper second order system

The equation of motion for the mass spring damper system shown in Figure 12 can be seen in the following equation: .[30], [31]

$$m\frac{(d^2x)}{dt^2} + b\frac{dx}{dt} + kx = 0 \tag{12}$$

The characteristic roots of Equation 12 can be found using the following equation:

$$r_1, r_2 = \frac{-b \pm \sqrt{b^2 - 4mk}}{2m} \tag{13}$$

It can be seen that the roots, or system poles, can take on different values according to the value of the square root in Equation 13. The roots can also be used to find the damping ratio, ζ according to following equation.

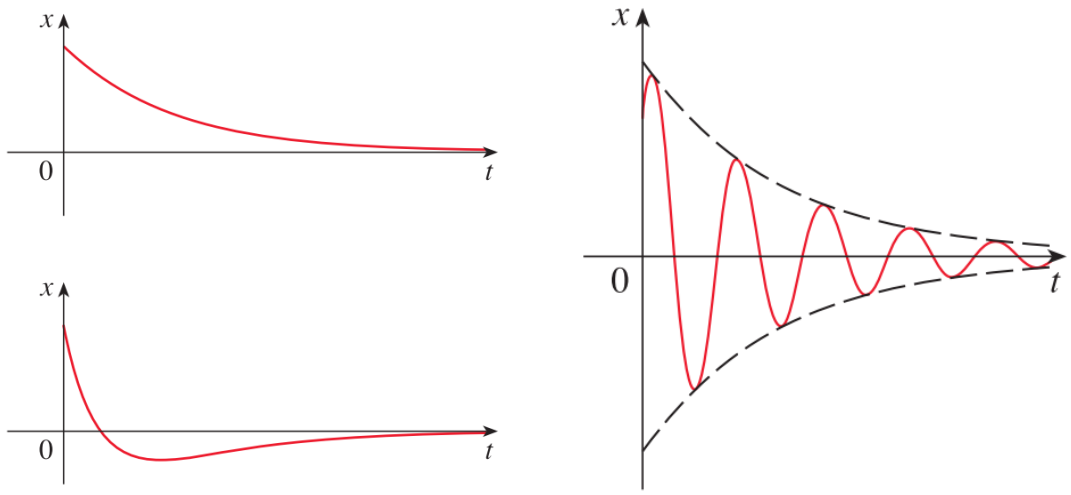
$$\zeta = \frac{b}{2\sqrt{km}} \tag{14}$$

The damping ratio is the ratio of actual damping to the critical damping. The different cases that cause a system to be underdamped, critically damped, or overdamped are described in Table 1.

Table 1. Damping conditions based on root values

Damping Condition	**Root Characterization**	**Value of Square Root in Equation 13**	**Damping Ratio (ζ)**
Underdamped	Non-real complex roots	$b^2 < 4mk$	$0 \leq \zeta < 1$
Critically damped	Repeated Real Roots	$b^2 = 4mk$	$\zeta = 1$
Overdamped	Distinct real roots	$b^2 > 4mk$	$\zeta > 1$

Typically in an underdamped system, oscillations are present. This is due to the pole locations, as previously discussed in section 1.5.1.1. A critically damped system is the desired system response. It has just enough damping to prevent oscillations and will quickly reach the steady state value. An overdamped system will also not have any oscillations. It is possible for overshoot to occur in an overdamped response, but oscillation will not occur. Figure 13 shows typical responses for both critically, overdamped, and underdamped systems.



(a) Overdamped/Critically Damped (b) Underdamped

Figure 13. System Responses for over, under, and critically damped systems [30]

1.5.2 System Identification

System identification is a tool that can be used to find the relationship between the system input and output. It is often necessary to control dynamic systems because it is very difficult to have all the necessary information about the environment and operating conditions to make a perfect control strategy before the system is in its working environment. [32] Figure 15, from T.C. Hsia, shows a block diagram representation of system identification. To gain information about the system, different measurable inputs are given to the system. However, as previously discussed there is always noise associated with measured inputs and outputs which can lead to inaccurate assessments. To deal with the noise, a statistical approach is taken, such as a least squares approach, to minimize the error of the resultant model. [33]–[35]

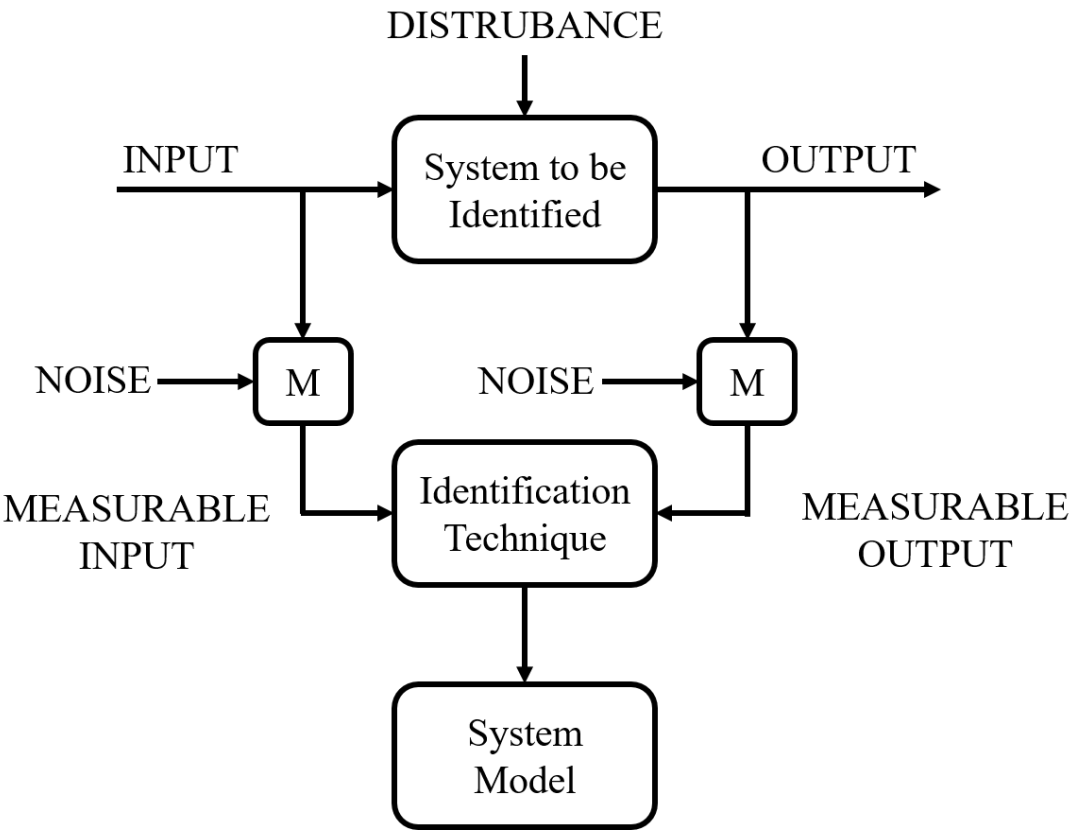


Figure 14. Representation of System Identification [33]

There are two main approaches to system identification. The first of which is a black box approach. This is used when nothing is known about the system. For example, if the order of the system is unknown and there is no dynamic model of the system, a black box approach will be used. The second approach can be described as a partial identification problem, or gray box modeling. With this setup, some characteristics of the system are known, such as linearity. However the values of the coefficients of a dynamic equation or the order of a dynamic equation may be unknown. This is an easier problem to solve than a black box problem.

1.5.3 Signals Used for System Identification

There are different types of signals that can be used for system identification. It is good to pick a signal which can excite the dynamics of a system throughout testing. Common choices for input signals include filtered white noise, pseudorandom signals, and binary signals, and random gaussian signals.

1.5.3.1 Pseudo-Random Binary Signal

A pseudo-random binary sequence (PRBS) has many characteristics similar to that of white noise. It can be generated according to the difference equation, seen in Equation 15. [34] It is best to use the PRBS as an input signal if the settling time of the system is not long.

$$u(t) = rem(A(q)u(t), 2) = rem(a_1u(t-1) + \cdots + a_nu(t-n), 2) \qquad (15)$$

1.5.4 Quantifying Model Fit

It is common to assign a model a value to represent how well the model approximates that data that it is attempting to predict. This can normally be quantified by the normalized root mean square error (NRMSE).

$$fit = \left(1 - \frac{|x_{ref} - x|}{|x_{ref} - mean(x_{ref})|}\right) \times 100\% \qquad (16)$$

where x is the predicted value, x_{ref} is the reference data that the model is compared against, and N_s is the total number of samples in the x and x_{ref} vectors. [36] A higher percentage of fit means a model is better at predicting the system output. However, it is important not

to over-fit a model during model creation. [33] This can lead to a bad fit against validation data.

1.5.5 Feed Forward Controls

Feed forward controllers do not use error-based information to command the system, but rather they use models to predict the system state and control. [26] Many times, feed forward controllers are used in conjunction with feedback controls to account for disturbances that may impact the system. They can also be used when it is not possible to obtain feedback from the system in real time.

1.5.6 Quantifying Controller Performance

The purpose of the controllers used in this experiment was to maintain a constant bead geometry. Therefore to quantify how good the controller performed, the mean absolute error (MAE) and root mean squared error (RMSE) was used to see how much variation was present within the bead geometry. The MAE can be found according to the following equation

$$MAE = \frac{1}{n}\sum_{j=1}^{n}\left|y_j - \hat{y}_j\right| \tag{17}$$

where n is the total number of samples, y_j is the current sample, and $\hat{y}_j$ is the average value of all the samples. [37] The RMSE could be found according to the following equation

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}\left(y_j - \hat{y}_j\right)^2} \tag{18}$$

It can be seen that the RMSE will generally penalize outliers more than the MAE because it squares the error term, while the MAE will not penalize outliers as much. Therefore, it can be useful to look at both the MAE and the RMSE when seeing how uniform a series of data points is.

1.6 Previous Work on Extruder Dynamics

A lot of work has been done to describe the steady state flow of material through an extruder. However, this work assumes a steady state process, Newtonian flow, and isothermal incompressible fluids are present in the extruder. [38] Nick Schott set up an experiment in 1971 to come up with a model for extruder dynamics by looking at an extruder's response to different inputs. To measure the flow out of the extruder, a ribbon was melted into the extrusion as it exited the die. The amount of ribbon melted in was recorded to measure how fast the material was extruded. Then, one foot sections of the extrudate were cut and weighed to find the mass flow rate of the extruder because it was known how long it took to extrude one foot sections. The extruder was found to have a second order underdamped response to a step change in screw speed. This study also used a black box approach and did not come up with a result that is generally applicable to any extruder. The system used to measure flow rate in the experiments performed for this thesis can be used on any extruder to learn about the system which is advantageous.

Other studies were performed looking at extrusion dynamics and surge with polystyrene. [39], [40] As with the previous study, parameter estimation and fine tuning of the process needed to be performed on a specific extruder to get good results. These experiments also looked at the longer term dynamics of the system. These previous experiments also do not deal with how the dynamics of the system change when being extruded onto a surface. This is important for LSAM because the way that the bead interacts with its environment will change the perceived dynamics of the system.

CHAPTER 2. EXPERIMENT SETUP

This chapter presents the experimental setup used to measure the flow rate out the nozzle. It explains the mechanical setup used to measure the flow rate out of the nozzle, as well as the instrumentation used to measure certain extruder properties. The chapter also presents the software setup that was used to drive the experiments and some problems with the experimental setup that needed to be addressed in order to obtain good data.

2.1 Machine Overview

To gain a better understanding of the volumetric flow rate out of the extruder, a bead characterization system (BCS) was created. As seen in Figure 15, the BCS features a 254.762 mm (10.03”) diameter heated spinning wheel which the polymer bead was extruded onto, a laser profilometer to measure the bead geometry, and a scraper to scrape material off of the wheel to allow for material to be extruded onto the wheel for an infinite period of time. The wheel spinning simulates the feed-rate during a normal print.

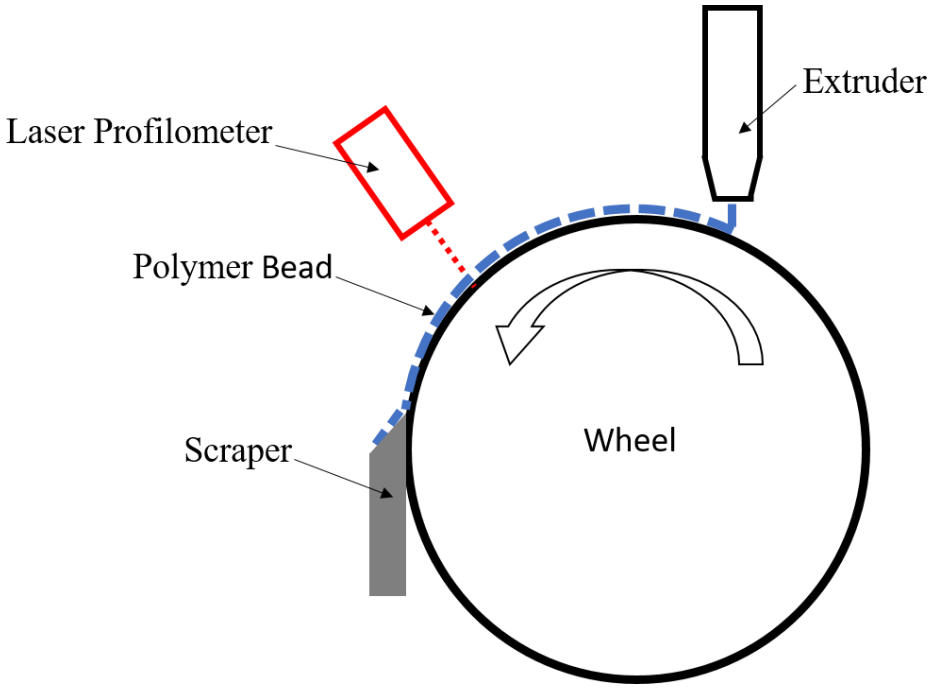


Figure 15. This is a visual representation of the BCS experiment setup.

The BCS was used on the Blue Gantry Machine (BG) at the Manufacturing Demonstration Facility (MDF) at Oak Ridge National Laboratory (ORNL). The BG is a BAAM system which uses a Dohley extruder, which was originally intended to be used for glue application. All experiments utilized a 0.2" diameter nozzle and used 20% Carbon Fiber ABS as the polymer. Data were collected with experiments known as runs.

Each run started with the extruder off, next, a series of voltage commands were sent to the extruder. The test concluded with the extruder off again. For each run, the wheel speed followed different patterns based on user selection through a Graphical User Interface (GUI) described in more detail in Section 2.3.1. The wheel could either move at a constant rotational velocity, spin proportionately to the extruder command, or be sent commands from a predictive controller which is presented in CHAPTER 4.

2.2 Instrumentation

Multiple sensors were used to monitor the system and gather the necessary data to calculate the flow rate out of the extruder. A quadrature encoder with 1024 pulses per revolution (ppr) was used to report the velocity of the extruder screw. This encoder can be seen in Figure 16 mounted to the extruder motor. Another quadrature encoder with 1024 ppr was used to track the absolute position of the wheel which was important for relating the time the polymer was extruded to the time it was measured. Each pulse of the encoder correlates to 0.038mm (0.0015”) on the surface of the wheel.

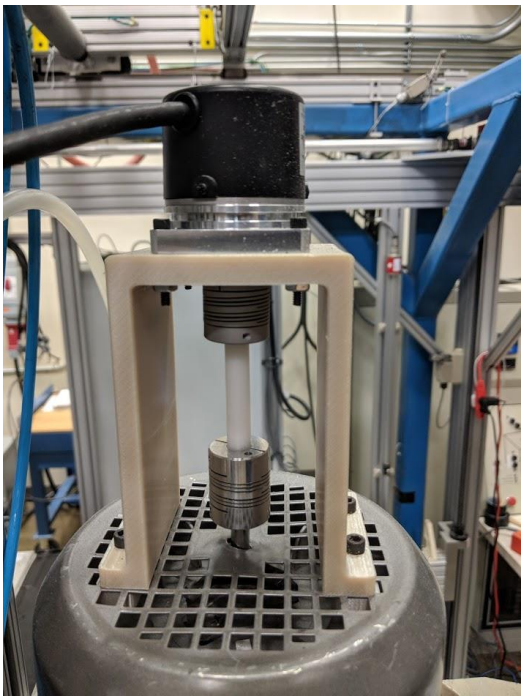


Figure 16. TRDA-20R1N1024VD Encoder mounted to the extruder motor.

2.2.1 Laser Profilometer

A laser profilometer was used to collect data about the bead geometry once it was printed on the wheel. The ScanControl 2610-50SI model was used, which can be seen in Figure 17. The laser profilometer was configured to continuously shine the laser and take

measurements at a frequency of 200 Hz. If the wheel is moving at 15 RPM, this corresponds to a measurement once per mm of bead extruded. To take a measurement of the bead, the laser had an onboard camera which took a picture of the laser against the surface of the wheel. Then it employed an internal CPU to perform image processing and transmitted the measurement to a LabVIEW system, which is discussed further in Section 2.3.2. For the pictures taken by the laser scanner, it is recommended to have an exposure of 70-90%. [41] The laser scanner was configured to have an exposure of 80%, as reported by the Micro-Epsilon software. This is the provided software to configure the laser.

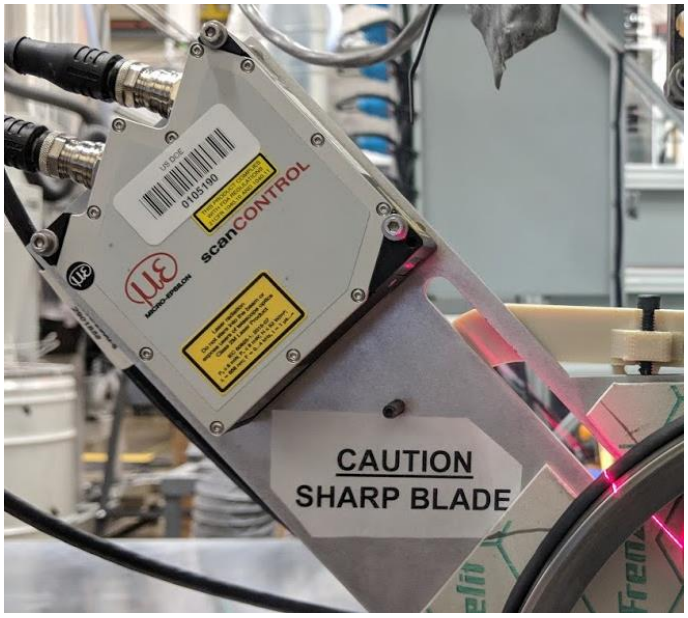


Figure 17. ScanControl 2610-50SI Laser profilometer shining on the bead

The laser was setup to use two reference areas, seen in Figure 18, to create a reference line to measure the bead against. After setting the reference areas, the laser does not consider any data points found outside of the reference area. This can be seen by the ignored data in the figure. It is important to set the reference zones toward the outside edges of the wheel

so that the bead geometry, seen in blue, did not interfere with those zones to keep a consistent reference line from which to take measurements.

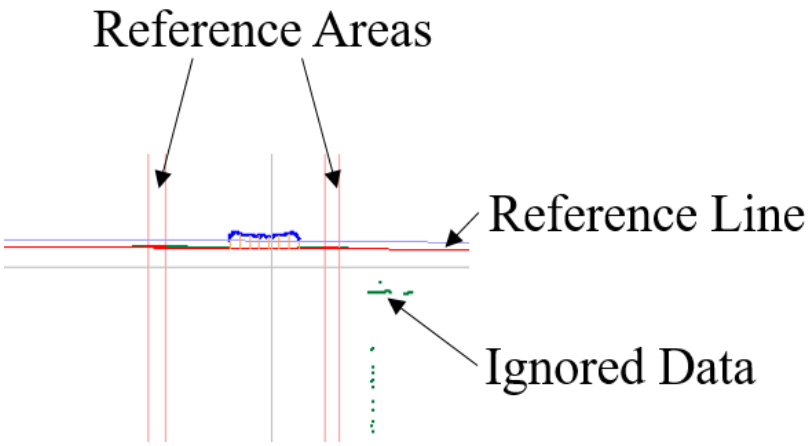


Figure 18. ScanControl software setup

2.3 Experiment Coordination

The BCS was built with the intent to be able to move to different machines enabling the capability to characterize different extruders in the future. As such it was necessary to coordinate tasks between a LabVIEW system, which was used for data acquisition, and the BG controller. The BG controller uses a Simulink Real Time controller to move the gantry and send signals to the extruder. Figure 19 shows the tasks addressed by each of the different systems, and the details of each system are explained further throughout the chapter. It can be seen that the LabVIEW system received data from the Simulink Real Time System and the BG. It also recorded the information in a data file. In addition, the Simulink Real Time System sent commands to the BG to control different aspects of the machine, such as the extruder and wheel.

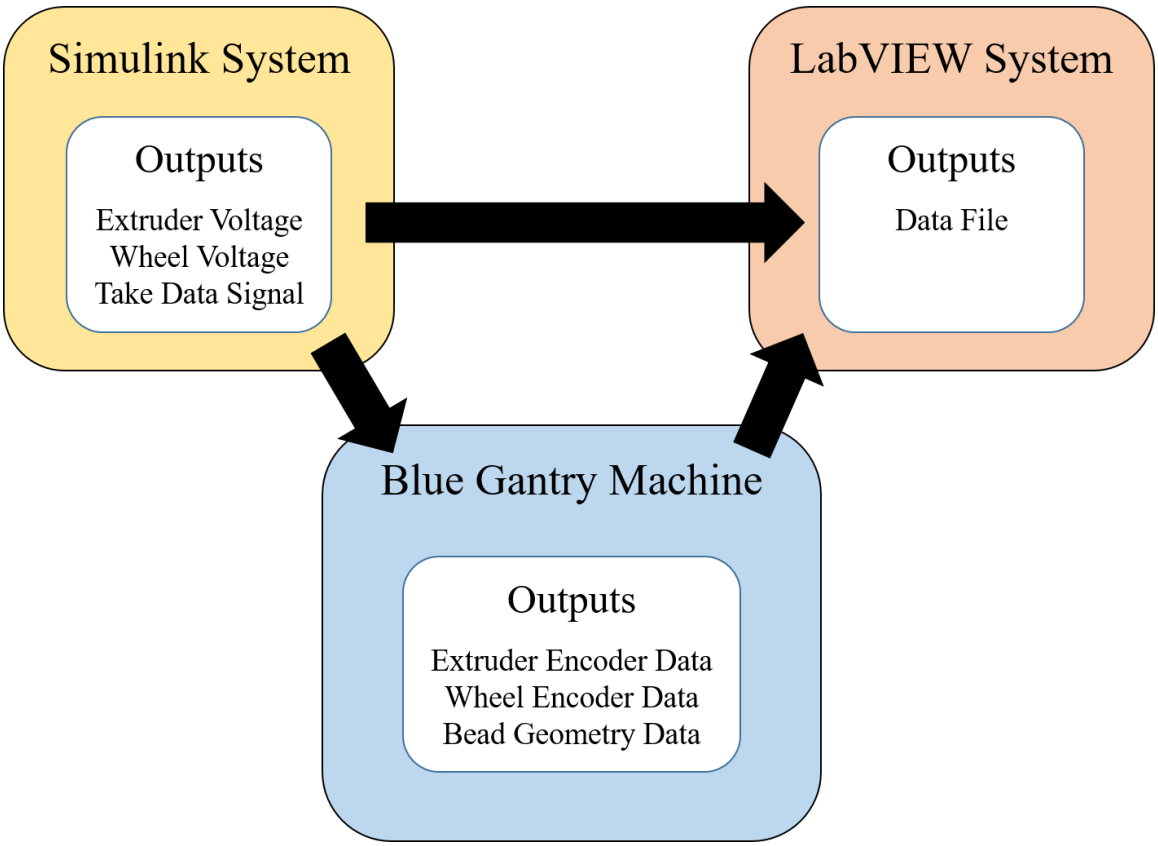


Figure 19. System Input and Outputs

2.3.1 Simulink Real Time Controller Modifications

To run tests on the BG, modifications needed to be made to the Simulink Real Time system. To facilitate ease of use, these features were bundled into a system identification graphical user interface (SIGUI), which can be seen in Figure 20. This SIGUI allows for multiple test types to be performed by changing the voltage sent to the extruder and the wheel command.

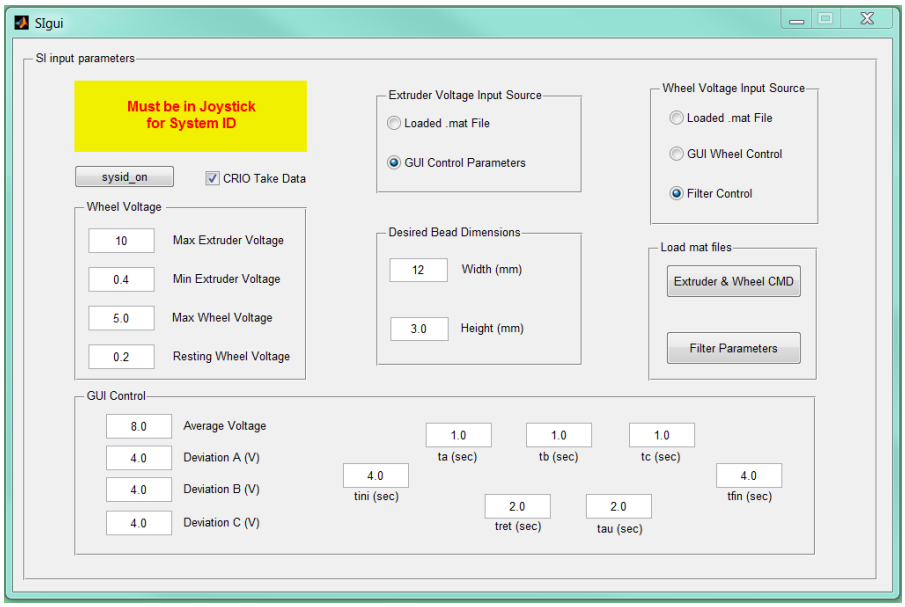


Figure 20. GUI to control the commands sent to the BG machine.

2.3.1.1 Extruder Commands

There were two main options for sending commands to the extruder that are explained throughout this section. First, the user could load a properly formatted file into the MATLAB command window. This enables complex profiles to excite the dynamics of the system to be created offline which reduced the amount of time spent on the machine during testing. An example of one of these profiles can be seen in Figure 21. The .mat file included two vectors: one vector which gave the desired voltage to send to the extruder, and a second vector which gave the transition times to move onto the next extruder command. The values in the transition time vector are the times at which the green lines intersect the x axis in Figure 21.

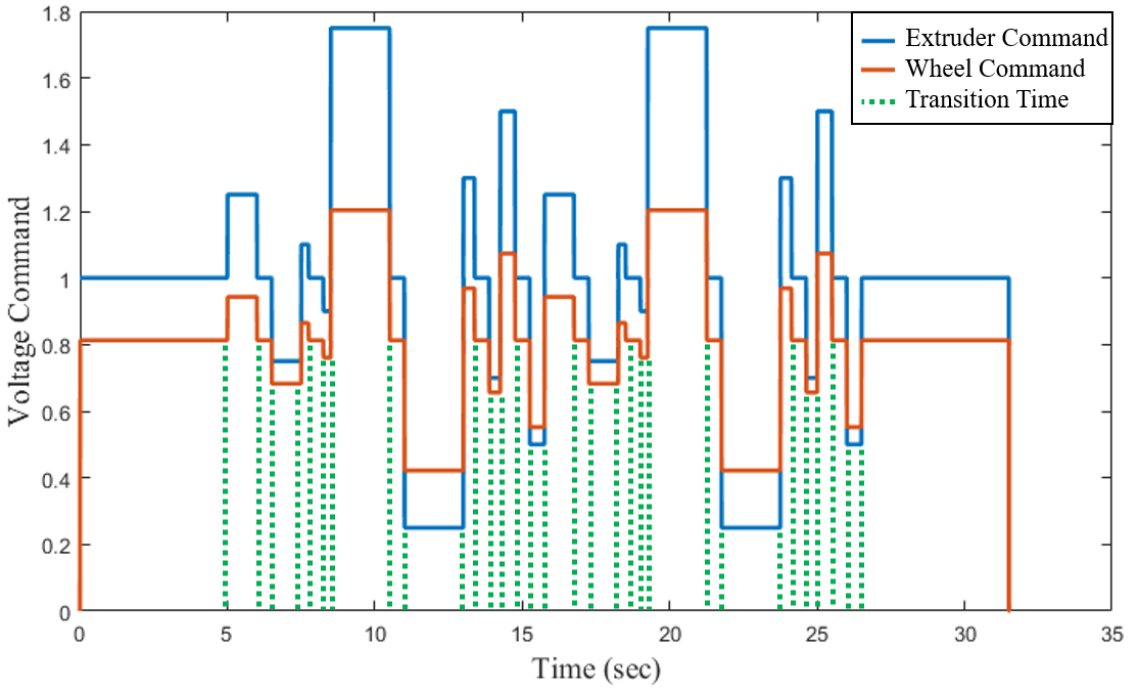


Figure 21. Sample Profile Created In MATLAB

The GUI also allowed the user to create a profile of extruder voltages using the "GUI Control" parameters seen at the bottom of Figure 20. These parameters created a voltage profile according to that of Figure 22. It can be seen that this profile consisted of three deviations from an average voltage, Deviation A, Deviation B, and Deviation C. The deviations last for times t_a, t_b, and t_c respectively. Each deviation occurred in both a positive and negative deviation from the average voltage. It was also possible to modify the time between each set of deviations, the time between the positive and negative fluctuation from the average voltage, as well as the time spent at the average voltage at the beginning and end of the sequence.

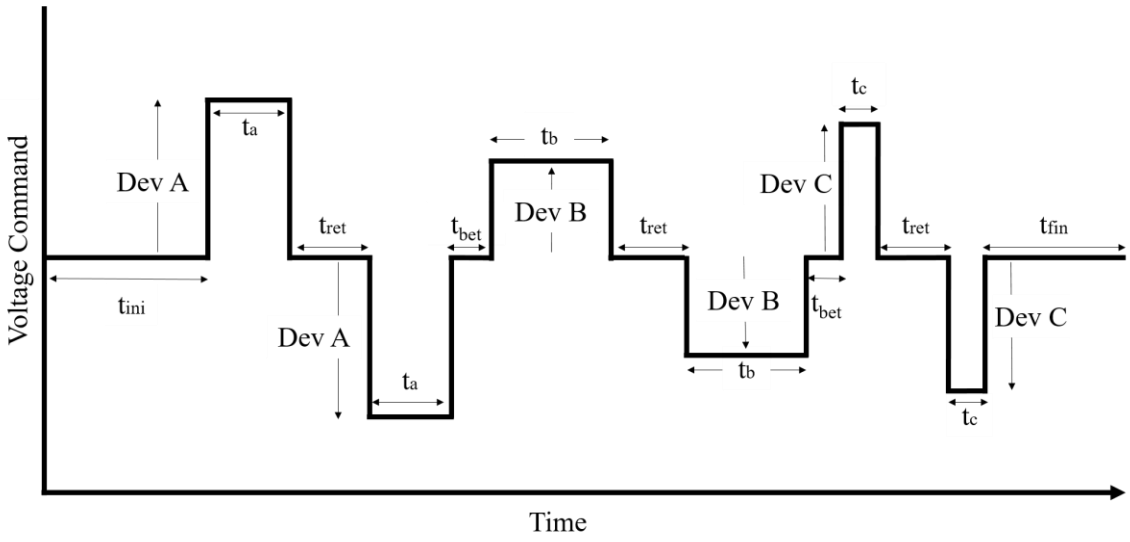


Figure 22. Voltage pattern created by the GUI Control parameters

2.3.1.2 Wheel Commands

The SIGUI has different options for the wheel speed. It could interpolate the command sent to the wheel based on the current extruder voltage through user selected interpolation values from the "Wheel Voltage" parameters, seen in Figure 20. A wheel command profile could also be loaded through a .mat file similar to the extruder profile.

Once the SIGUI had a model of the system built in from the system identification techniques discussed further in Section 4.1, it was possible to change the model coefficients through the SIGUI to test different models. This model was an option to command the wheel voltage sent to the system in coordination with the "Desired Bead Dimensions" to have a feed forward controller.

2.3.2 LabVIEW System

The LabVIEW system served as the data acquisition unit. It received inputs from the Simulink Real Time system and the sensors and was able to record the data in text files. Part of the front panel for user input during testing can be seen in Figure 23. This was used to view information, such as the bead dimensions and wheel speed as the experiments were performed. It also served as an interface to turn the sensors and actuators on.

As previously mentioned, this interface was responsible for saving data to text files to process after the experiments were conducted. There were two options for saving data to a file. First, there was a button, “CRIOSave”, that can be seen in Figure 23 which was responsible for saving data when activated. It was also possible to make the BG machine responsible for saving. In this case, saving is activated in the SIGUI which was discussed previously in this chapter. When a run was started, the SIGUI forced the BG to send a signal to the LabVIEW system to start saving data.

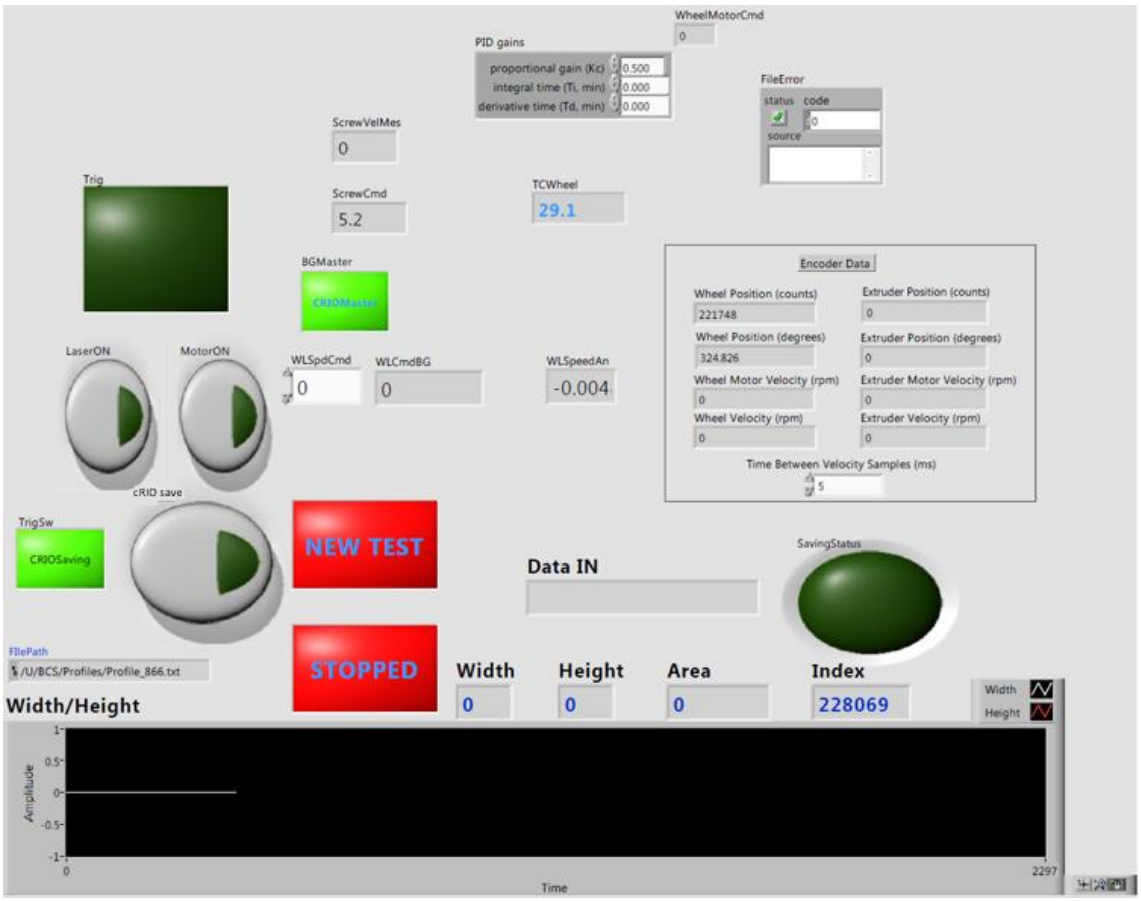
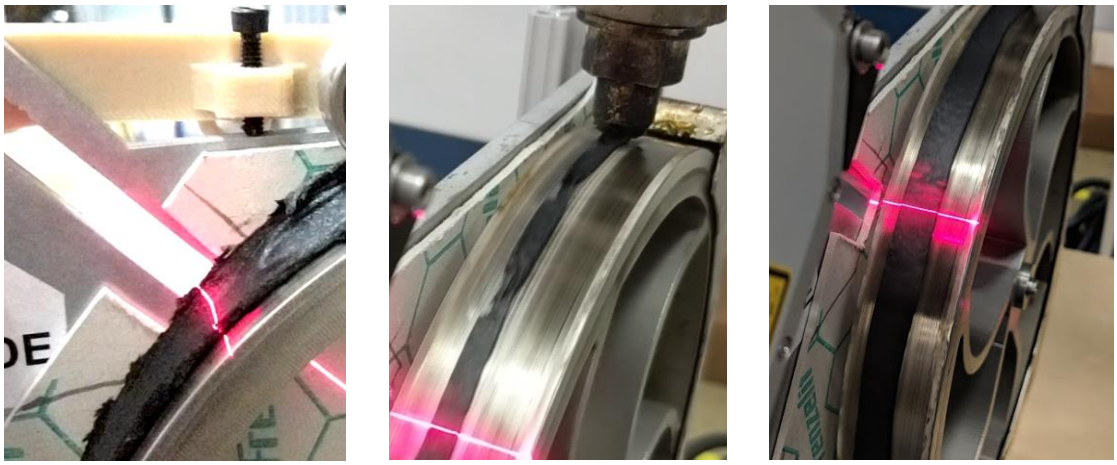


Figure 23. LabVIEW Front Panel

2.4 Difficulties with Experimental Setup

The BCS needed fine tuning to ensure accurate measurements of the system were taken. Problems specifically arose with the polymer bead quality on the wheel, which was largely due to the wheel speed relative to the extrusion speed. Figure 24a, b, and c show beads formed at wheel speeds that are too slow, too fast and a good wheel speed respectively. If the wheel was moving too slow then the polymer would bulge out and have a bad surface, seen in Figure 24a. This made it difficult to both measure with the laser and properly correlate to the time extruded, which is discussed in Section 3.1.1. If the wheel was rotating too fast relative to the extrusion rate, as seen in Figure 24b, the polymer tore and did not

stick to the wheel. Figure 24c shows a clean bead profile due to the relationship between the extrusion rate and wheel speed. A trial and error approach was used to find wheel speeds that allowed the polymer to easily stick to the wheel and produce good bead geometries.



(a) Wheel Moving Too Slow (b) Wheel Moving Too Fast (c) Good Wheel Speed

Figure 24. Bead properties with different wheel speeds

Different problems arose with starting and stopping the extrusion. When starting to extrude, there is a time delay between the time the screw starts extruding and the time material begins to come out of the nozzle because the shear field is not fully developed within the extruder barrel. This time delay varies based on the residence time, or how long the extruder had been sitting idle. Therefore synchronization of the wheel speed and extrusion rate is difficult. As such, there were significant problems to get the polymer to stick to the wheel in the beginning of a run.

To combat this problem different approaches were taken, of which two methods saw some success. The first of which involved applying Cube Glue to the wheel before starting a run.

This helped to adhere the polymer to the wheel. Cube Glue is an adhesive glue that comes with some FDM 3D printers to apply to the print bed before starting a print to help the material stick to the build platform. Another method that saw some success with polymer adhesion to the wheel was ramping the wheel speed to try to match the rate of extrusion. This method saw some success, but the ramping speed necessary to get a good bead profile for measurement changed with the polymer's residence time in the extruder and the extruder command.

There were also problems with adhesion to the wheel when turning the extruder off. After the extruder is sent a command to stop, there is still "drool" that comes out of the nozzle. The amount of "drool" depends on the extruder command before turning the nozzle off. If the wheel speed at the end of the run was too fast, the polymer would hit the wheel and curl up next to the nozzle as shown in Figure 25. If the wheel moves too slow during the period of time when the "drool" is exiting the nozzle, a bead similar to that shown in Figure 24a will occur causing it to be difficult to calculate the flow rate out of the nozzle.

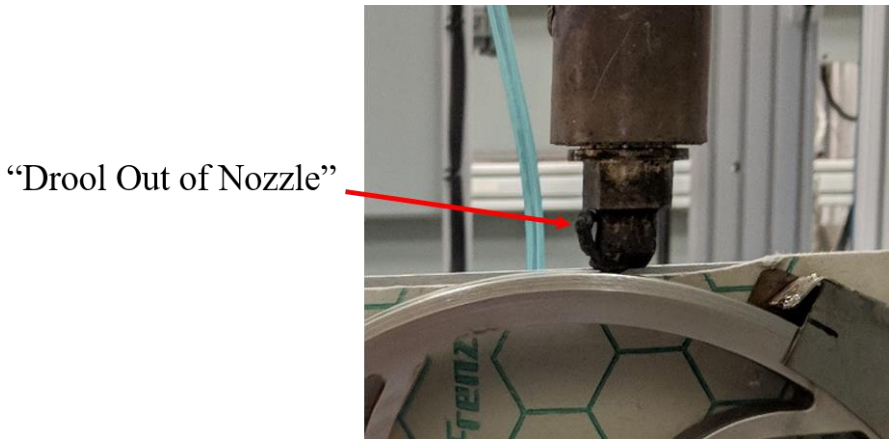


Figure 25. Wheel moves too fast causing "drool" to curl up.

CHAPTER 3. DATA AND ANALYSIS

This chapter presents how the data was processed after the experiments that were described in the previous chapter were performed. First how the volumetric flow rate was calculated is discussed, then methods used to filter the data are presented, and finally the model created to represent extruder dynamics is described.

3.1 Calculating the Volumetric Flow Rate

3.1.1 Finding the Encoder Offset

Due to the setup of the BCS, there is an inherent time delay between the time the polymer is extruded and the time the bead is measured by the laser profilometer. This time delay changes in conjunction with the wheel speed throughout the course of a run, but the absolute position between the point of extrusion and the point of measurement is always constant. This absolute position offset was known as the encoder offset for the purposes of this experiment. The encoder offset is the number of encoder pulses between the point of polymer extrusion and the point of measurement.

It is important to have an accurate number of encoder pulses for the encoder offset value to ensure that the flow rate calculation was accurate for lining up the data, which is discussed in the next section. To find the encoder offset, a pin was placed under the middle of the nozzle, and the encoder position in this state was noted. Figure 26 shows the setup to find the encoder offset. Once the pin was placed, the wheel was rotated until pin passed under the laser.

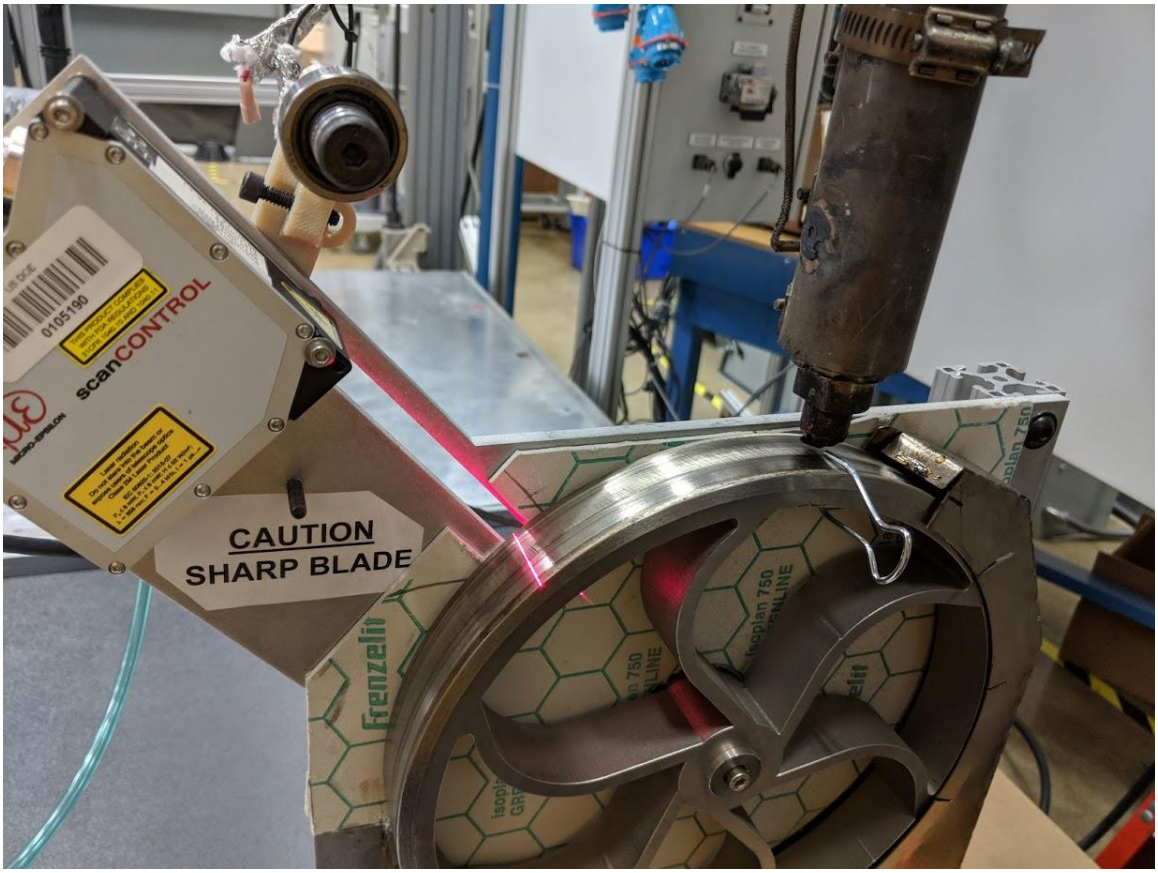


Figure 26. Pin placed under nozzle to find the encoder offset.

The encoder offset was calculated according to Equation 19,

$$\Delta p_{enc} = p_{max} - p_{start} \tag{19}$$

where Δp_{enc} is the encoder offset, p_{max} is the encoder position at the time of the maximum measurement from the laser profilometer, and p_{start} is the starting position of the encoder. The encoder offset procedure using the pin was repeated 6 times. This would result in a range of encoder offsets. After all the encoder offsets were found, the average value was taken. This wound up with a range of positions within 1.524mm (0.06”) which is still within the nozzle. The encoder offset procedure was performed every time data was taken

to account for the position of the BCS changing relative to the extruder position. There were errors introduced because a person placed the pin each time, which would lead to slightly different measurements.

3.1.2 Measurement Time and Extruder Time

When the data was recorded, the time the material was extruded was not the same as the time the material was measured. As such, once the encoder offset was found, it was necessary to align the time each measurement was taken with the laser profilometer to the time that the material was extruded using the extruder offset.

To line up the data, an algorithm was created. The basic premise of the algorithm can be seen in Table 2. The goal of the algorithm is to find the bead dimensions in extruder time, where extruder time is defined as the dimensions of the bead at the time the material was extruded. The table shows "X" in positions where the values are not important for this example.

The algorithm looks at the current encoder position, pointed out in step one in the table. The encoder offset is added to each of the encoder positions. To find the Desired Encoder Position which is pointed out in box two of the table. For this example the encoder offset was 10,000 clicks. The algorithm looks ahead in the data to find where the number of encoder clicks lies, seen in step three. In this example, the exact number of encoder counts desired could not be found. So it finds the two data points that the Desired Encoder Position lies between. In step 4, the algorithm looks at the measured bead dimensions at 0.235 and 0.240 seconds. Finally, the algorithm performes interpolation to find the bead area, width and height in extruder time, seen in step 5.

Table 2. Basic Algorithm Structure for Lining up Time Extruded to Time Measured

Time	Encoder Counts	Desired Encoder Position	Measured Bead Dimensions	Bead Dimensions Extruder Time
0	2568910	2578910	X	X
0.005	2568942	2578942	X	Area: 12.5 mm² Width: 10 mm Height: 2.5 mm
● ● ●	● ● ●	● ● ●	● ● ●	● ● ●
0.235	2578930	X	Area: 12.5 mm² Width: 10 mm Height: 2.5 mm	X
0.240	2578950	X	Area: 12.5 mm² Width: 10 mm Height: 2.5 mm	X

1 2 3 4 5

3.1.3 Calculating the Volumetric Flow Rate

Once the data were processed, the flow rate was calculated according to Equation 20,

$$Q = \omega \cdot A_{bead} \tag{20}$$

where Q is the volumetric flow rate, ω is the velocity of the wheel surface in mm/s at the time of extrusion, and A_{bead} is the cross sectional area of the bead in extruder time, as found by the algorithm that was explained in the previous section.

3.2 Data Processing

As discussed in Section 1.4, noise is always present in systems. As such, there was noise present in signals received by the LabVIEW system. To fix some of the disturbances, ferrite cores similar to the one shown in Figure 27, were added to the wiring to reduce interference from high frequency fluctuations in wires that had PWM signals passed through them. In

addition, the grounding of all the electrical components was checked to ensure that there were no ground loops present. These noise prevention techniques reduced the noise in the analog signals, such as the voltage command sent to the extruder and wheel. However, there was still noise present in the measurements received from the laser profilometer and the wheel speed, given by the encoder on the wheel.

Figure 27. Ferrite core used to limit interference. [42]

To make the data easier to process, the cross sectional area of the bead as reported by the laser profilometer and wheel velocity measurements were filtered before calculating the flow rate. The data were filtered using a hampel filter to eliminate outliers and a Savitsky-Golay filter was used to smooth the data.

Figure 28 shows the bead width and wheel velocity measurements before and after filtering. It can be seen from this figure that the overall trend from the data still stayed the same. In the wheel speed data, it can be seen that sometimes the encoder does not report a wheel velocity. The wheel did not actually stop rotating during the experiments. The Hampel filter successfully worked to eliminate these outliers from the data.

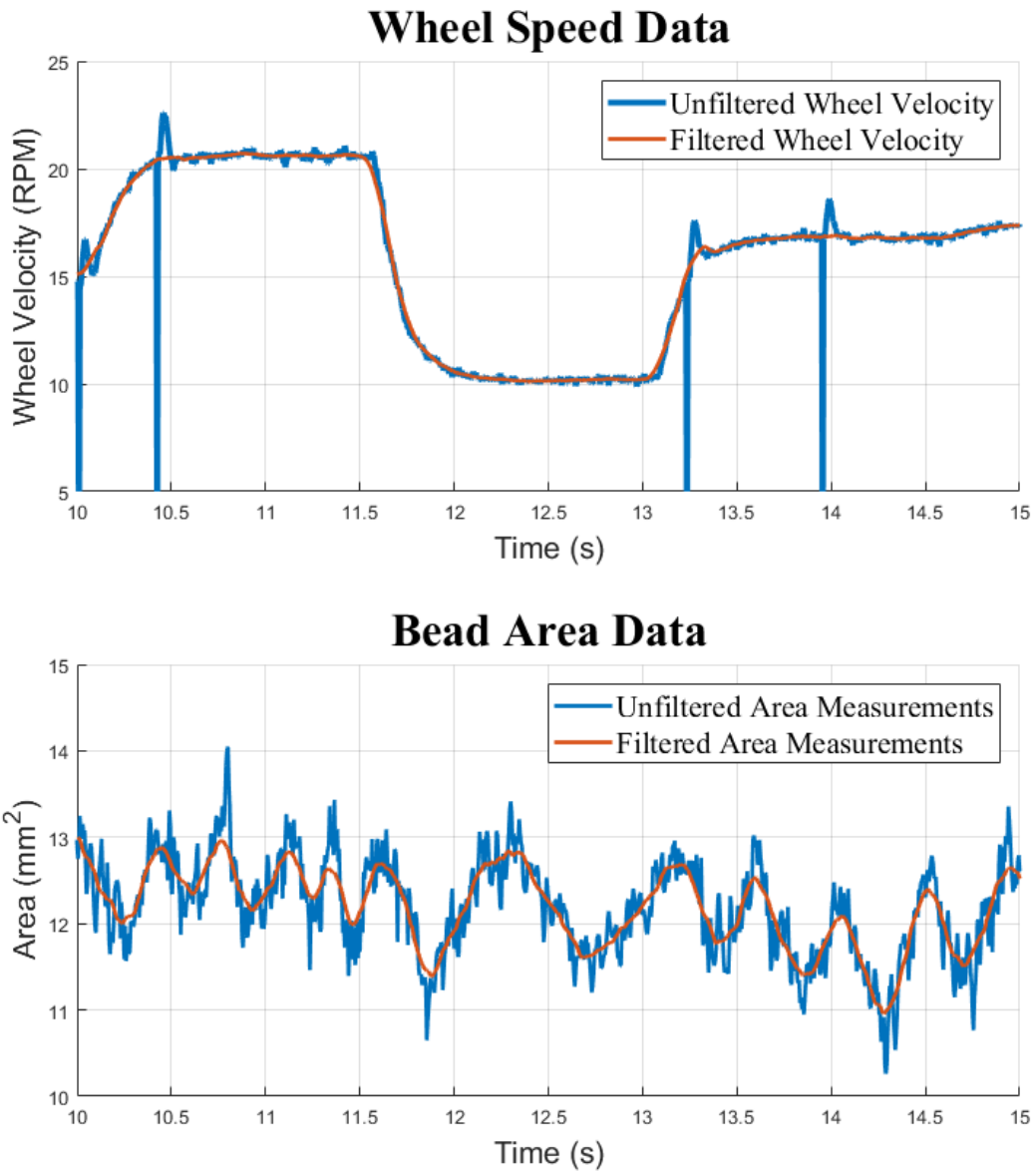


Figure 28. Data Before and After Filtering Run 869

3.3 Modeling the system

To model the system, the MATLAB System Identification Toolbox was used. This toolbox has many different methods of finding the transfer function between an input and an output. In order to make a good controller, a model for each stage of the system was created. The

system was modeled in a black box modeling approach because it is difficult to create an equation to describe the dynamics of the extruder.

3.3.1 Extruder Voltage to Extruder RPM

The model used to describe the commanded extruder voltage to the actual extruder RPM was a second order model with one zero. The model shows an overdamped response for the extruder, which is to be expected for a motor. The poles were located along the real axis at -7.43 and -0.09, and the zero was located at -0.08. It can be seen that this is close to a first order system, as one of the poles and zeros almost cancel each other out, but a first order system did not do as good of a job at properly defining the system, even though it is how a motor theoretically should respond. The motor was controlled by a variable frequency drive which could have led to some variation from the first order system. The model also had a gain of 35.08 and a time delay of 0.054 seconds. The transfer function for the commanded extruder voltage to the extruder RPM, y, can be seen in Equation 21.

$$y = \frac{439s + 35.08}{1.523s^2 + 11.44s + 1} \cdot e^{-0.054s} \tag{21}$$

Figure 29 shows the model validated against validation data, which were not used to create the model. The top of the figure shows the extruder command, and the bottom of the figure shows the actual response in red and the predicted response in blue. As seen in the figure, the predicted extruder velocity closely matches actual motor velocity. The NRMSE of the run shown is 90.42%.

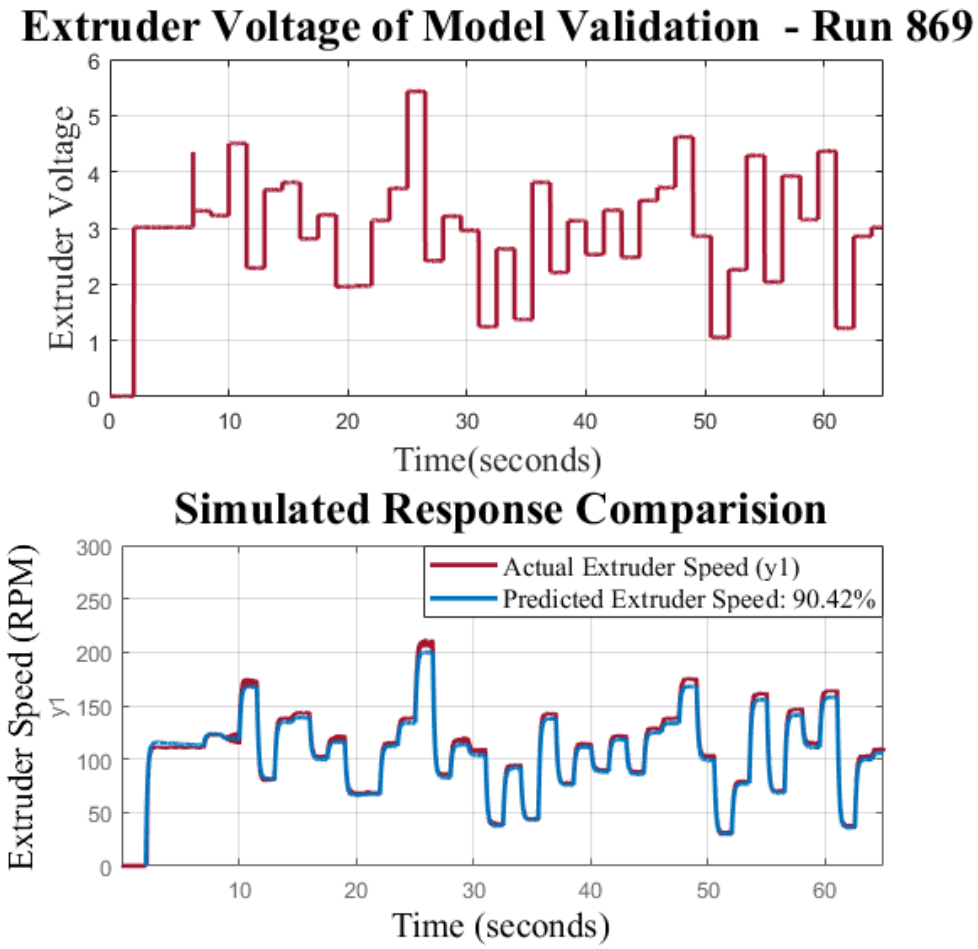


Figure 29. System Identification Extruder Voltage to Extruder Speed

3.3.2 Extruder RPM to Volumetric Flow Rate

A model was created to relate the volumetric flow rate out of the nozzle to the extruder RPM. To choose the best model, the system identification toolbox in MATLAB was used to perform system identification with the data from different runs. Models were created using Run 831 as the basis for model creation. Run 831 used a PRBS as the commanded extruder voltage. Next, the models created were validated against other runs. In other words, the models were fed the input RPM from the run and then compared to the actual flow rates from the runs. Table 3 shows various order models created from Run 831 being

validated against another run, Run 821. It is best to have as simple of a model as possible from a controls perspective, so it is disadvantageous to add more poles and zeros.

Table 3. Fit for Different Models Validated Against Run 821

Number of Poles	Number of Zeros	NRMSE (%)
1	0	57.45
1	1	63.39
2	0	76.78
2	1	78.52
2	2	80.99
3	0	76.64
3	1	78.41
3	2	80.56

As seen in Table 3, the model with two poles and two zeros had the best fit during validation. The poles and zeros for this model are both complex conjugates with the poles at $-0.738 \pm 0.993i$ and the zeros at $-0.0619 \pm 0.1108i$. The poles are complex conjugates, indicating that this is an underdamped system.

As previously mentioned, the model was created from data collected from run 831, which used a PRBS as the input signal to the extruder. Figure 30 shows the model created between the extruder RPM and the calculated flow rate. In the graph on the bottom, the simulated response can be seen in blue and the calculated response for the run can be seen in red. It can be seen that the model does not pick up the higher frequency fluctuations, but it does pick up the overall trend of the data. The NRMSE is given as 82.6% for this model, where a value of 100% would be a perfect fit to the data.

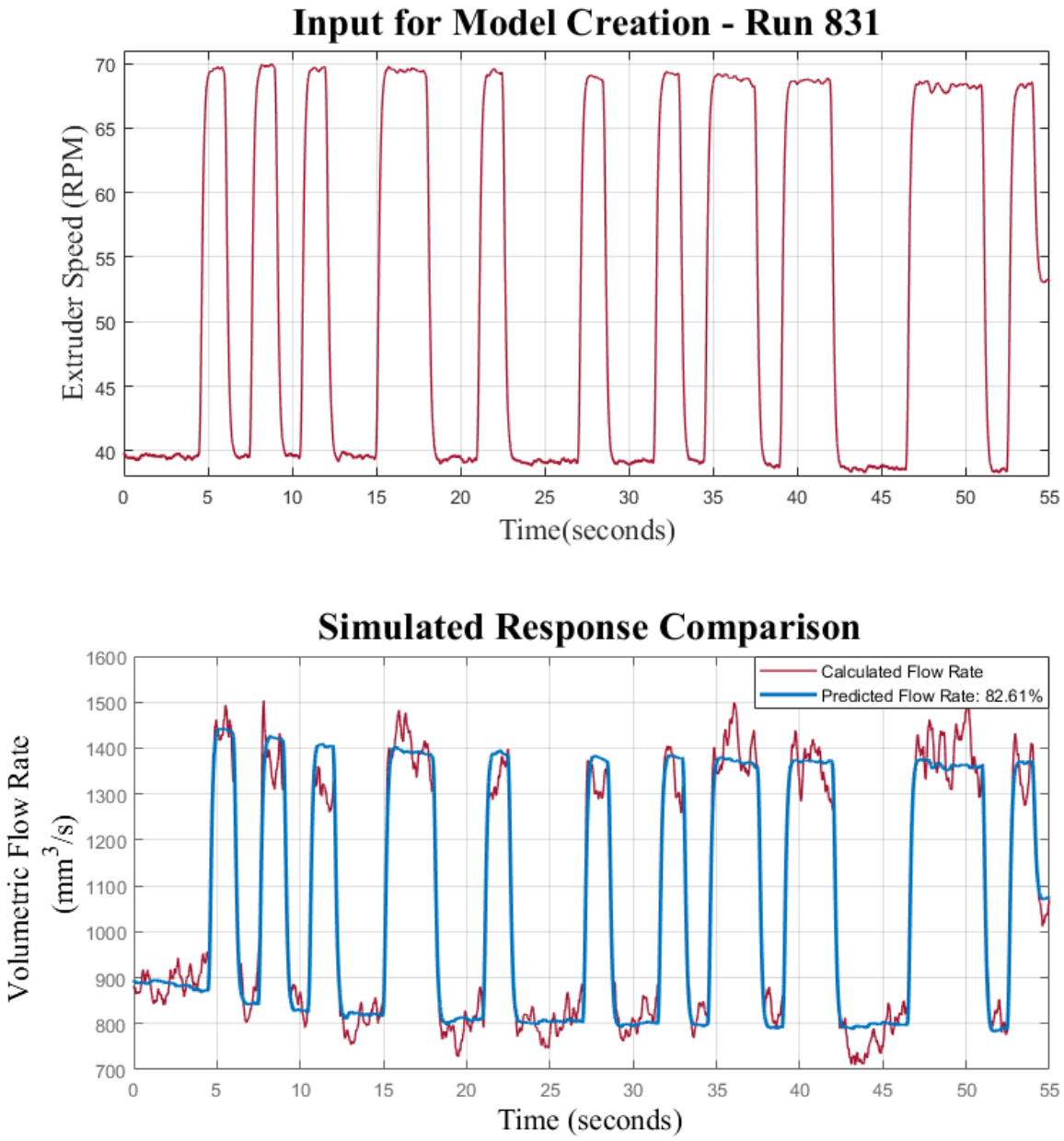


Figure 30. Model Creation for Extruder RPM to Volumetric Flow Rate

After the model shown in Figure 30 was created, it was validated against other runs. One of these validation runs can be seen in Figure 31. The top of the figure shows the input RPM for the validation data, and the bottom of the figure shows the calculated flow rate in red and the predicted flow rate in blue. It can be seen that the model runs through the

calculated flow rate for most of the run, but it does not capture the higher frequency changes in flow rate. However, this model provides a metric for tracking the transitions between the different extruder speeds. The model has an 83.26% NRMSE fit to this calculated flow rate.

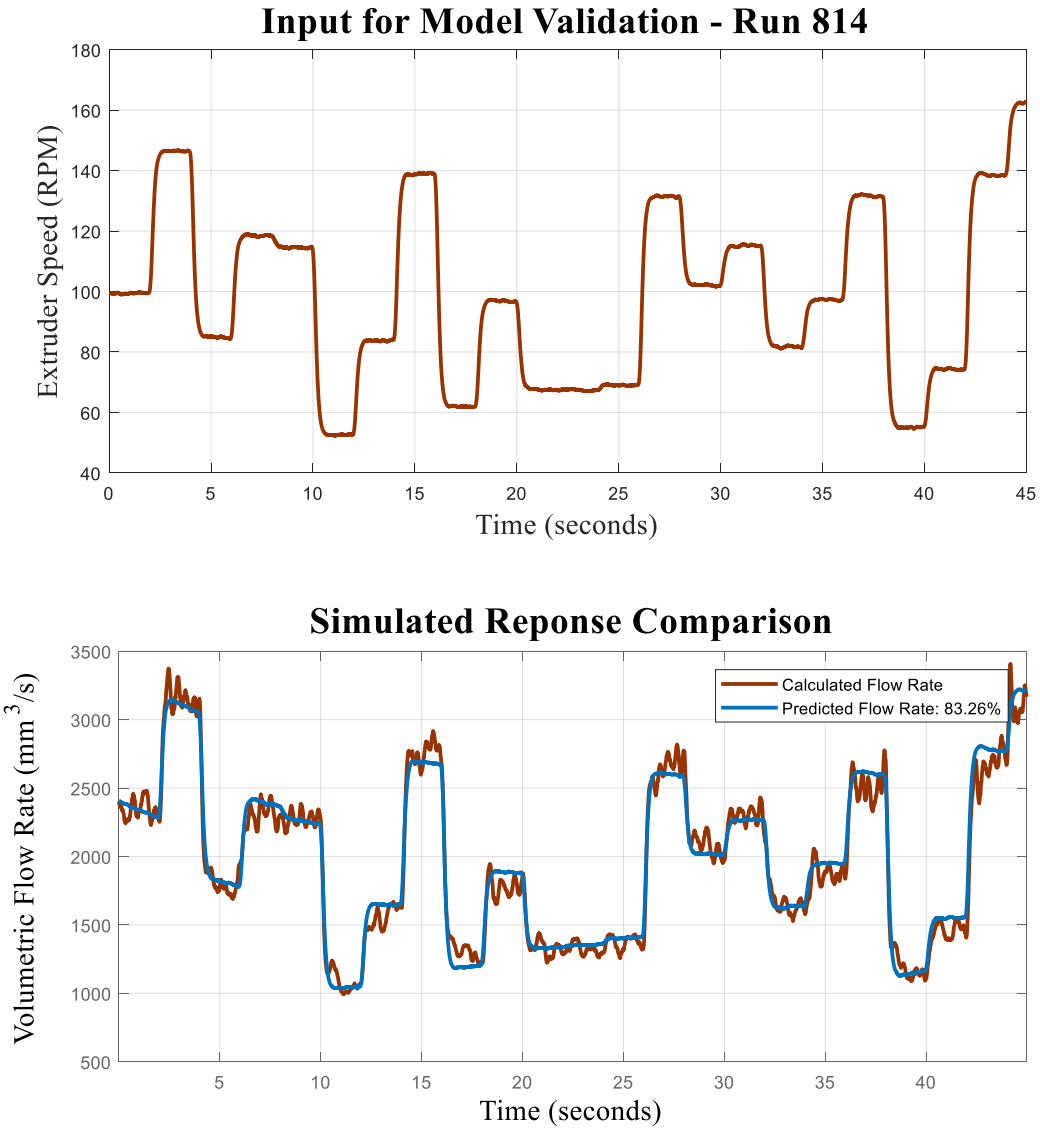


Figure 31. Validation Run for Chosen Model

3.3.3 Wheel Voltage to Wheel RPM

Another model was created to serve as a transition between the commanded voltage sent to the wheel motor and the wheel speed. Figure 32 shows the model creation and validation for the commanded wheel voltage to the actual wheel speed in RPM. It can be seen that this motor was very predictable with validation data having a 95.86% NRMSE. The model used for the relationship between the commanded motor voltage and the wheel RPM was a first order model with a time delay. The values for dc gain, pole, and time delay were 2.517, 0.0377, and 0.009 respectively.

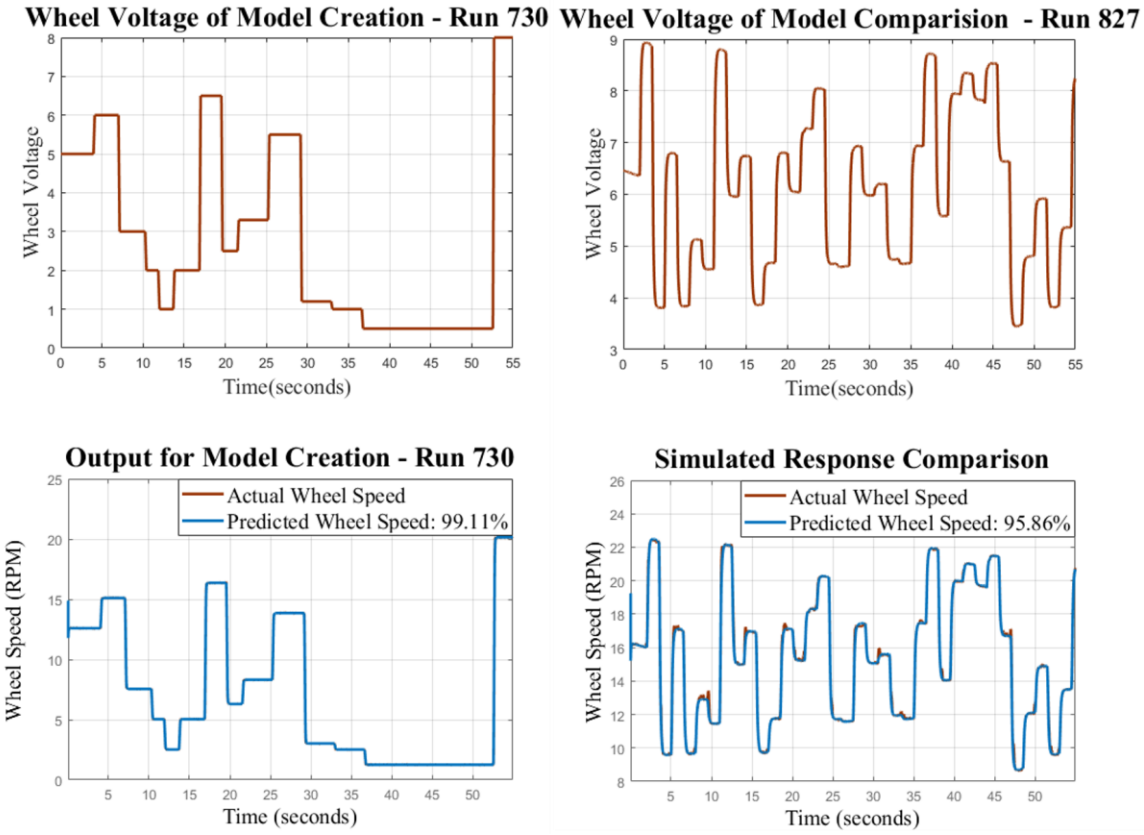


Figure 32. Wheel Speed Model Creation and Validation

CHAPTER 4. CONTROLLER CREATION AND RESULTS

This chapter shows how the model created in the previous chapter was used to control the wheel speed to maintain a constant bead geometry. First, the architecture for the model is presented. Next, the controller implementation is shown, and then the results from runs with and without the controller enabled are presented. Finally, sources of error are discussed.

4.1 System Model for Controller

To control the wheel speed to correspond to the flow rate out of the nozzle, a system model was necessary. Figure 33 shows the overall model for the BCS. It can be seen that the system takes in the current extruder command. It uses transfer function G1 to find the extruder RPM from the model explained in Section 3.3.1. The transfer function that was obtained in Section 3.3.2, G2, is applied to the output from G1 to find the predicted volumetric flow rate given the current command. The system uses the user defined geometry from the SIGUI to calculate an area of the bead. This is used to find the necessary wheel surface speed to maintain a constant bead geometry. Finally, the model was multiplied by GC, the controller transfer function. This was created by using pole placement to make the wheel respond more appropriately to user commands. GC placed the modeled transfer function pole from the wheel command to wheel velocity in the numerator and put a new pole in place of it to get a more controlled response.

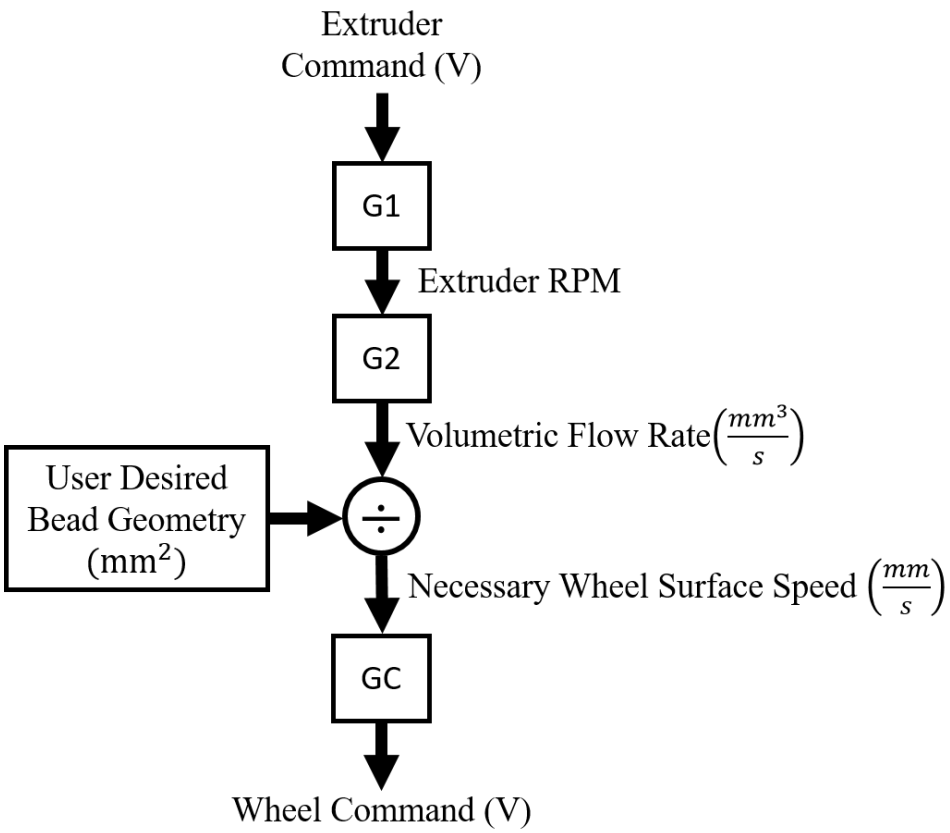


Figure 33. BCS Controller Model

The model created was only valid for the middle of an extrusion pattern. In other words, the beginning and ending of the extrusion was not modeled. The dynamics at play when the extruder is turned on and off are different than the dynamics when in the middle of an extrusion. This is an area of further research that needs to be done.

4.2 Controller Implementation

All of the models were created in the continuous time domain because the deposition process is continuous. However, the controller implementation used discrete time because the Simulink Real Time system sends commands to the BG machine in discrete time at a

frequency of 200 Hz. Therefore the transfer functions were converted from continuous time to discrete time in MATLAB.

4.2.1 Wheel Controller

The wheel speed was manipulated to maintain a constant bead geometry. To obtain a better response from the wheel a control block was added into the system which had a zero to cancel the modelled pole of the system and a pole was placed closer to the origin at 0.01 for a more controlled response.

4.2.2 Wheel Versus Extruder Control

The feed forward controller implemented varied the wheel speed to maintain a constant bead width. This is roughly equivalent to changing the feed-rate, or the speed at which the gantry moves, during printing operations. The wheel speed was changed relative to the commanded extruder voltage, because the wheel has a faster response time than the extruder. This made it easier to control because the wheel was always able to respond fast enough to keep up with the changes in the flow rate.

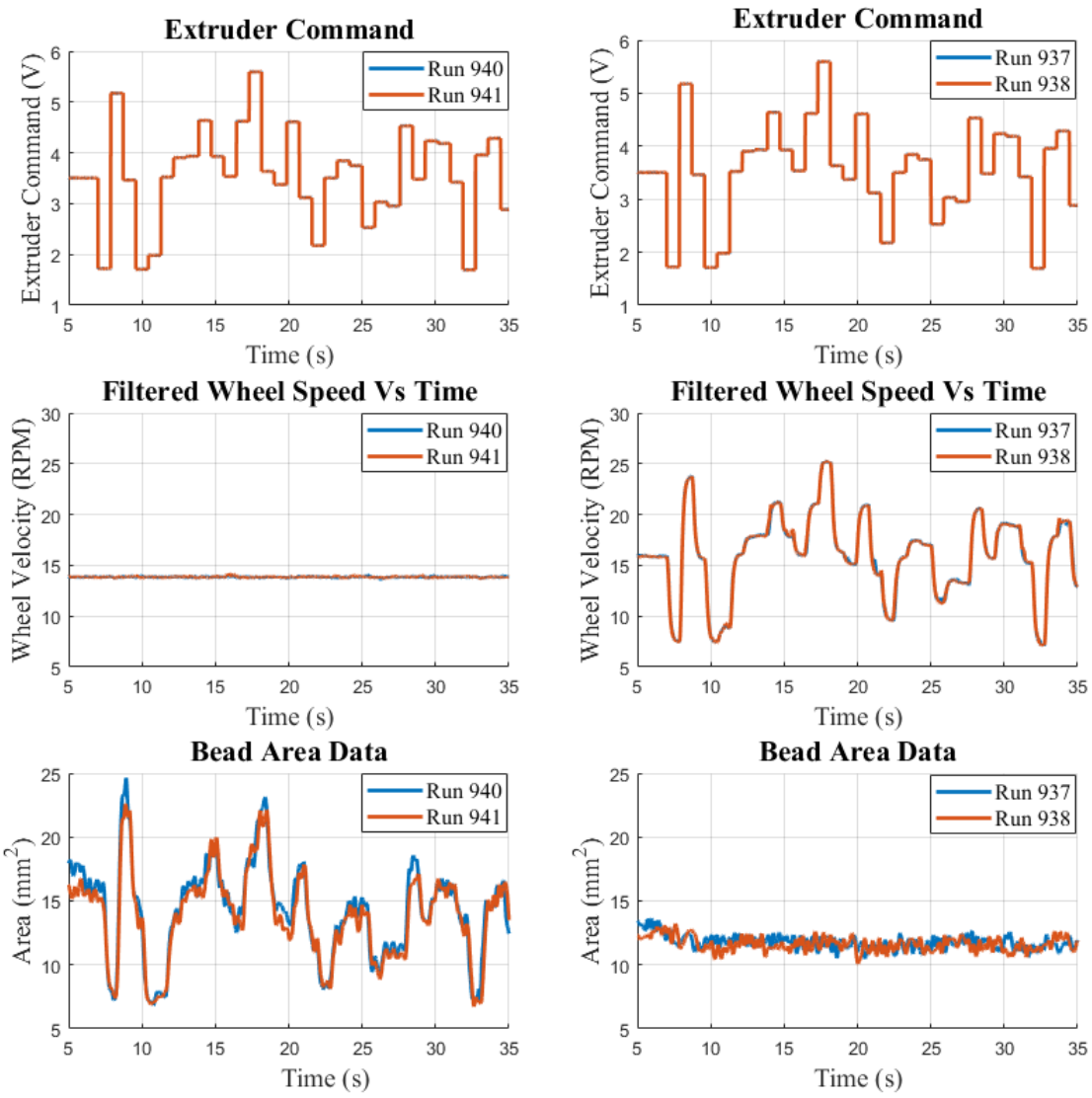
If a profile for wheel speeds was given instead of extruder commands, the controller would need to limit the acceleration of the wheel to allow for the extruder to compensate. When a controller is implemented in the future to control the bead width while printing, changes will be made to both the feed-rate and the extruder command. Ideally the feed-rate should never change and continue to be the commanded feed-rate from the G-code. However, it is likely that in many cases the flow rate will not be able to respond fast enough to keep up

with the change in the feed-rate so the feed-rate will need to be controlled in conjunction with the extruder command. This control scheme is further explained in section 5.3.

4.3 Controller Results

To test the effectiveness of the implemented controller, various extruder profiles were run with the controller on and off. When the controller was off, the wheel was sent a constant commanded voltage. The constant wheel speed was picked such that there was as little tearing of the bead as possible.

The results from one of the tests can be seen in Figure 34. For the series of tests shown, a random Gaussian signal was used as the extruder command. It can be seen that for the tests shown in Figure 34a, the commanded wheel speed was held constant, while for other runs, shown in Figure 34b, the controller was on and the wheel voltage command changed according to the controller output. Figure 34 also shows the bead area data from the runs with and without the controller.



(a) No Controller (b) Wheel Controller Implementation

Figure 34. Bead Area Fluctuation Without Control and with Wheel Controller

Upon taking a closer look at the data shown in the Bead Area Data of Figure 34b, it can be seen that there is a slight overall downward trend for the bead area over time. A possible explanation for this is that it takes more than 5 seconds to reach steady state when starting the extruder because the material printed for the first part of the print has sat in the extruder without moving before the experiment was run. This would lead to a different viscosity of

the material. Therefore, it may take more time for the shear field to fully develop to obtain a more predictable bead.

For these runs, the RMSE and MAE, were calculated from the average bead width and area for the runs with and without the wheel controller implemented, as seen in Table 4. It can be seen that there was about six times less variation from the average bead area with the controller enabled, and over 3.5 times less variation from the average bead width with the controller enabled.

Table 4. Error Comparison for Bead Geometry Controller

		RMSE	**MAE**
Area	**Controller On**	0.57 mm^2	0.44 mm^2
	Controller Off	3.43 mm^2	2.72 mm^2
Width	**Controller On**	0.49 mm	0.38 mm
	Controller Off	1.91 mm	1.53 mm

4.3.1 Controller Refinement

Interestingly while the controller produced good results, the DC gain was not quite tuned properly. In other words, the user specified bead geometry was not the produced geometry, which can be seen in Figure 35b. It is clear that when given a command of 3.5mm, that the actual average width of the bead was 13.1mm. To address this, a series of tests were performed to measure the bead geometry with different user inputs into the SIGUI, explained in section 2.3.1. For these tests, the wheel controller was always on, and different bead width commands were given.

Figure 35 shows some of the results from this test. It can be seen that there was variation in the bead width between different runs and inputs. Figure 35a also shows that the average bead area was much more consistent than the average bead width, as seen by the runs being more clustered. The tamper was not running during any of these tests, this led to the shape of the bead changing even though the overall area was more consistent. When printing a part this should not be a problem because the tamper knocks the bead down to a relatively constant height, which should result in a more consistent width. No tests were performed with the tamper on. This can be a source of further experimentation, and should be done to tune in the DC gain on the controller.

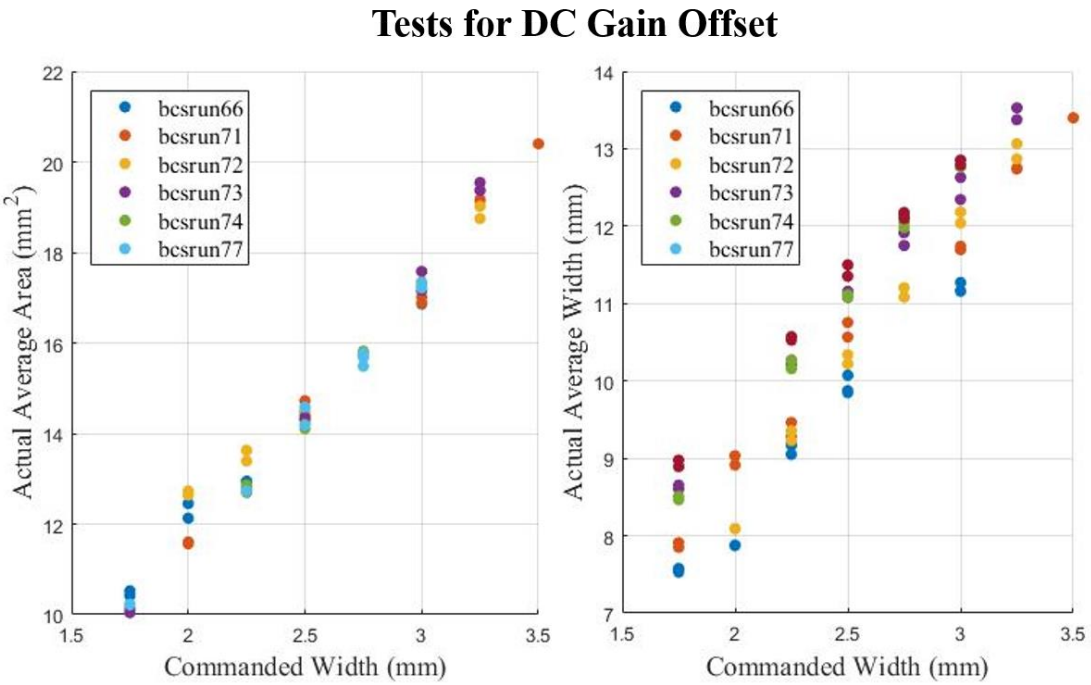


Figure 35. Test Results to Find the DC Gain

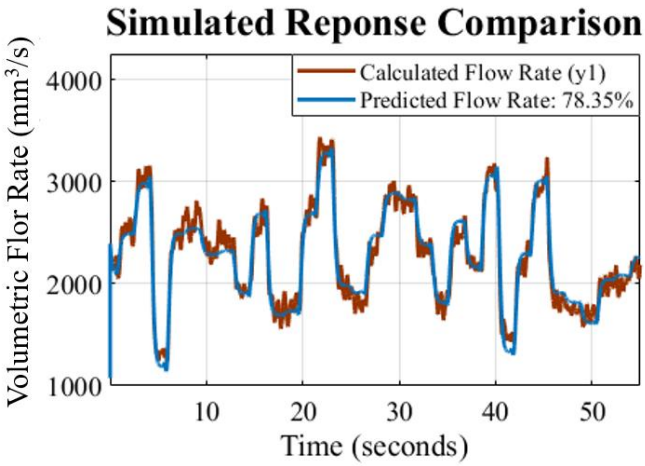
4.3.2 Different Results with a Constant Wheel Speed

When testing the controller, some runs were performed with a constant wheel speed. It became apparent that the polymer behaved differently under this condition. A new model was created for the polymer under this operating condition which can be seen in Equation 22.

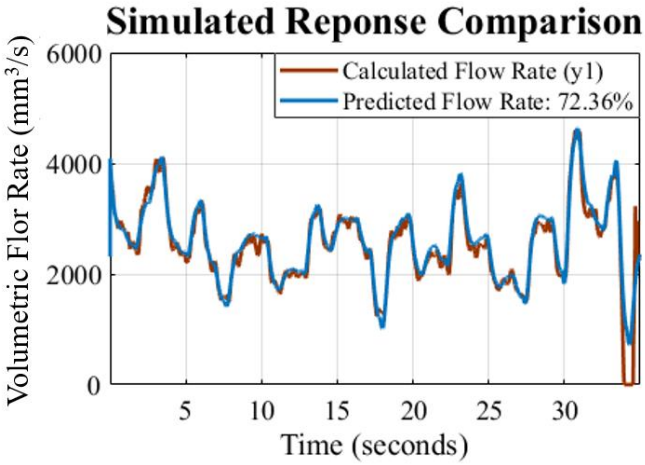
$$\frac{9.175s^2 - 149.2s + 905.3}{s^2 + 12.84s + 47.52} e^{-0.01s} \tag{22}$$

This has system poles at $-6.4198 \pm 2.5107i$ and system zeros at $8.1304 \pm 5.7062i$. It is clear that this is still an underdamped system, but now there is a time delay between the extruder velocity and the volumetric flow rate, which was not present before. A possible explanation for this is that the CF-ABS is a non-Newtonian material that behaves differently under shear conditions, and the wheel changing speeds was placing additional shear on the system.

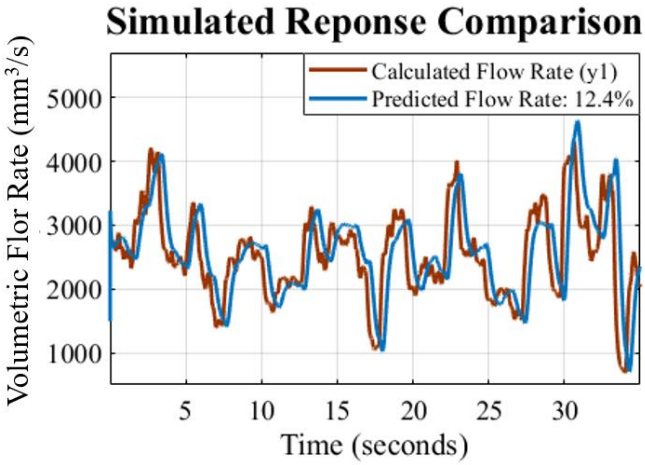
Figure 36 shows three different plots of the model created. Figure 36a shows the data used to create the model in red, and the models prediction in blue. Figure 36b shows the same model against validation data. It can be seen that the model has a fit, or NRMSE, of 72.36% against the validation data. This shows that this is not a bad prediction for a constant wheel speed. Figure 36c shows the model created for a constant wheel speed validated against data that was collected with a varying wheel speed. This makes it clear that the material behaves differently with a constant wheel speed when compared to a varying wheel speed.



(a) Model Creation for Constant Wheel Speed



(b) Model Validation for Constant Wheel Speed



(c) Validation agains run with non-constant wheel speed

Figure 36. Model Creation and Validation for Runs with and Without Constant Wheel Speed

4.4 Sources of Error

4.4.1 Encoder Offset Error

While the controller seemed to have promising results, the experiment was not completely without error. One of the largest sources of error was the encoder offset, discussed in Section 3.1.1. While the encoder offset was within the range of the extruder nozzle, it is unclear where exactly within the nozzle the real offset is. The measurements attempted to find the offset between the middle of the extruder nozzle and the point of measurement. However, it is possible that a better estimation of the flow rate would be obtained if the offset was measured from the closest or furthest point within the nozzle to the point of measurement.

4.4.2 Error Due to Adhesion

The BCS attempted to replicate printing conditions to measure the flow rate by using the wheel as a simulated feed-rate. To get better adhesion between the wheel and the polymer coming out of the nozzle, the tamper was turned off. The tamper is almost always used during a print to knock down bead geometry that is above the plane of the nozzle to avoid tearing of the part. Therefore, to generate a more accurate model of the bead, a system in which the tamper can be used may lead to a more accurate model.

4.4.3 Error Due to the Laser Profilometer

The laser profilometer operation was discussed in Section 2.2.1. This showed that the laser shined on the bead from above and had a camera take a picture of the laser against the bead geometry and then performed integration to find the bead area. Due to the setup, this

overestimated the bead area because part of the bead was obscured by the laser. Figure 37 shows a representation of the cross sectional area of a bead. On the left is a possible bead geometry that a laser is shining down on. It can be seen that the bead on the left has rounded corners on the bottom that the laser cannot see because it is shining from above. Therefore, the laser over approximates the area of the bead as seen in the measured bead on the right. Obstruction due to the beads geometry was present while printing and definitely led to error in the total material printed for a run.

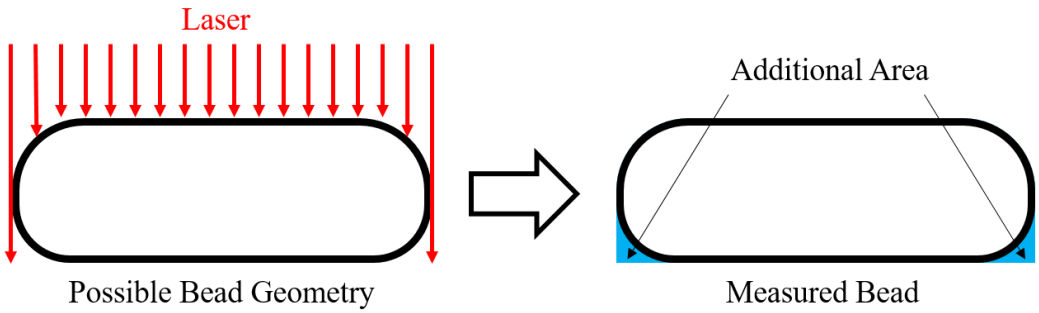


Figure 37. Error in Bead Measurement Due to Laser Profilometer

4.4.4 Variation Between Runs

While models were created from the collected data, that produced reasonable fits to validation data, there was variation between experimental runs. For example, Figure 38 shows the calculated flow rate for different runs which were all given the same extruder commands. It can be seen that there is variation between the runs. The total volume printed was calculated from the flow rate data for these runs and there was some variation. The average amount of material printed was 97400 mm^3, but the maximum amount of material printed occurred in Run 741 which printed 10400 mm^3 of material which is 7.74% larger than the mean, and Run 748 printed the least amount of material with 87900 mm^3 of

material, which is 9.75% less than the mean. This variation is a source of error in the experiment. One possible explanation of it is porosity in the material, which is discussed in the next section.

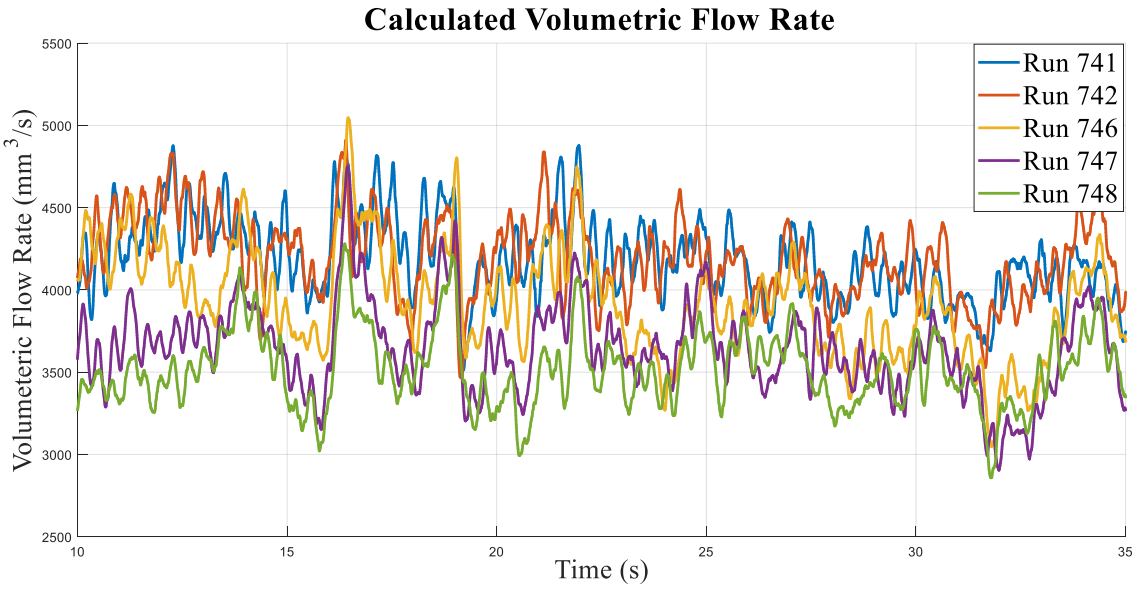


Figure 38. Variation in Flow Rate Between Runs

4.4.5 Porosity in the Material

Ideally the material that is printed is incompressible and has no porosity. However, the extruder used for this experiment is not an ideal extruder. It is recommended to use an extruder with at least a 20:1 L/D ratio, but the Dohley extruder used does not meet this ratio. This causes the printed material to have some air entrapped in the bead. This is a possible source of error that could lead to variations between experiments.

CHAPTER 5. CONCLUSIONS AND FUTURE WORK

The BCS seemed to show successful results, as shown by the results from the controller. However, more testing needs to be completed to ensure that the created models are correct and should be implemented on a machine controller in the future. This chapter is organized as follows: first, conclusion about the BCS are presented; then, further experiments that can be performed with the BCS are shown; finally, another experimental setup for collecting data about the extrusion dynamics is discussed.

5.1 Conclusions

This thesis presented a method for measuring the volumetric flow rate and implementing a controller to maintain a consistent bead geometry for LSAM. The BCS was used to measure the flow rate by simulating a feed-rate with a rotating wheel, and a laser profilometer was used to capture the bead geometry. Different profiles in the form of random Gaussian signals and PRBS were sent as extruder commands to excite the dynamics of the system. The results from these tests were then used to create models of different portions of the extruder system. The model used had two poles and two zeros. It yielded an over 80% fit against validation data. Next, a controller was implemented to maintain a constant bead geometry by varying the wheel speed to match the flow rate out of the nozzle. This controller offered great improvements over a constant wheel velocity. There was about six times less variation in bead area throughout the course of a run when the controller was enabled. The BCS proved to be a reliable way to measure the flow rate of a system. Other forms of additive manufacturing, such as metal additive systems, face similar issues with unknown dynamics. A similar methodology can be applied to other

systems to find system models of the deposition behavior to apply to the control schemes to allow for less part defects.

5.2 Further Experimentation with the BCS

As previously discussed in Section 4.3.2 the polymer behaved differently when there was a constant wheel speed versus when there was a changing wheel speed. Future experiments can be done with a volumetric flow rate model for data collected with the wheel moving at a constant velocity. It would be interesting to compare this controller to the current controller to see how much the dynamics between the bead and wheel impact the bead geometry.

Additional work can also be done with the BCS to characterize starting and stopping extrusion. Many of the part defects are due to the starting and stopping of the extruder, so a better understanding of the delay of material flow out of the nozzle during start up. One of the challenges that must be solved for this is reliable adhesion between the material and the wheel to avoid slip during start up. To test the effects off different residence times in the barrel, extruder commands with known time intervals of the extruder being turned off can be sent allowing for the effects of starting after different residence times can be measured.

All of the current runs have performed with the same nozzle diameter, 0.2”, and the same material, 20% CF-ABS. However, different nozzle diameters yield different extruder dynamics. For example, with a smaller nozzle diameter the pressures experienced during printing are much higher for the same throughput of material, which causes different dynamics. As such, there is “drool” coming out of the nozzle after turning it off for a longer

period of time. Additionally, each material behaves differently while printing due to different viscosities and compositions. To implement a controller in the machine to account for the extruder dynamics, different material and extruder properties will need to be known. Therefore, it is recommended that experiments with different materials and different nozzles be performed, and models must be created for each set of tests. A lookup table of different properties should then be created in the machine controller to address the material and nozzle properties during a print.

5.3 Future Model Validation and Control Implementation

5.3.1 Eliminating a Model from the Controller

Currently there are three models used to predict the necessary wheel behavior to maintain a constant bead width. Each of these models are discussed in Section 3.3. One of the models, G1, predicts the current extruder speed from the commanded voltage sent to the extruder. Since all models inherently have error, in the future the extruder velocity seen from the encoder should be sent back to the system to eliminate one prediction. This should allow for better control over the bead width.

5.3.2 Future Control Implementation

To test to see that the BCS has a good model of the extrusion dynamics, it is necessary to make a part using a controller created from the data collected. As previously discussed in Section 4.2.2, the control implementation will be different than when controlling the bead width on the BCS. This will require implementing the controller in the framework that the machine interprets G-Code. Figure 39 shows a block diagram of potential implementation

on a printer. G-code is an input to the system. It is used to set the machine parameters and tell the machine where material needs to be printed. The trajectory planner takes parsed G-code, and tells the machine how to move in order to reach the points along the path specified. The robot control takes the specified points and considers the limitations of the machine, such as acceleration limits. To improve upon the current control strategy of an CNC gantry, a machine dynamics controller that contains the models generated from experiments performed with the BCS will inform the way commands need to be sent to the printer. The machine dynamics controller will store the models created from the BCS so that the feed rate and expected flow rate can be coordinated to maintain a constant bead geometry throughout a print. For example, when traversing a corner the robot controller will change the path to decelerate.

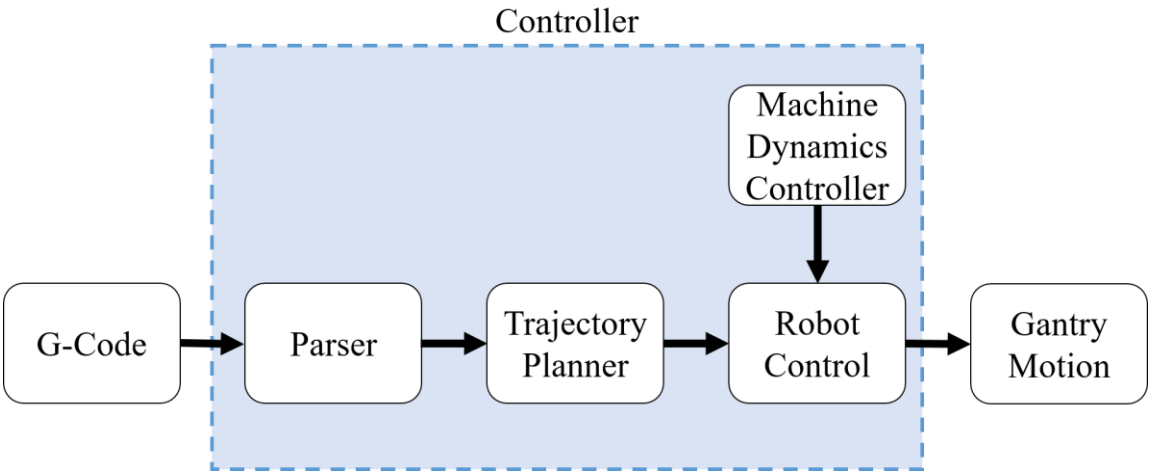


Figure 39. Controller Implementation on a Machine

5.4 Alternative Experiment Setup

The BCS is one approach that can be taken to measure the extruder dynamics. However, it is not a perfect system. The BCS attempts to replicate printing conditions by having the

material extruded onto a surface that is simulating the feed-rate during printing. To purely measure the extruder response to different commands, a different approach will need to be taken, because the BCS introduces different dynamics. Specifically, measuring the starting and stopping response of the extruder will be very difficult with the BCS due to getting the material to reliably stick to the wheel without slipping.

A proposed method for measuring the flow rate out of the nozzle is proposed in Figure 40. This method involves using a laser Doppler velocimeter (LDV) to measure the speed of the bead coming out of the nozzle. LDVs are commonly used in the cable industry to track cable movement during the manufacturing process. [43] A laser profilometer can be placed slightly above or below the LDV to obtain the profile of the bead. It is recommended that a ring style laser profilometer is used so that the entire bead profile can be measured. This will eliminate one source of error where part of the object that the laser is viewing is obscured in the current setup.

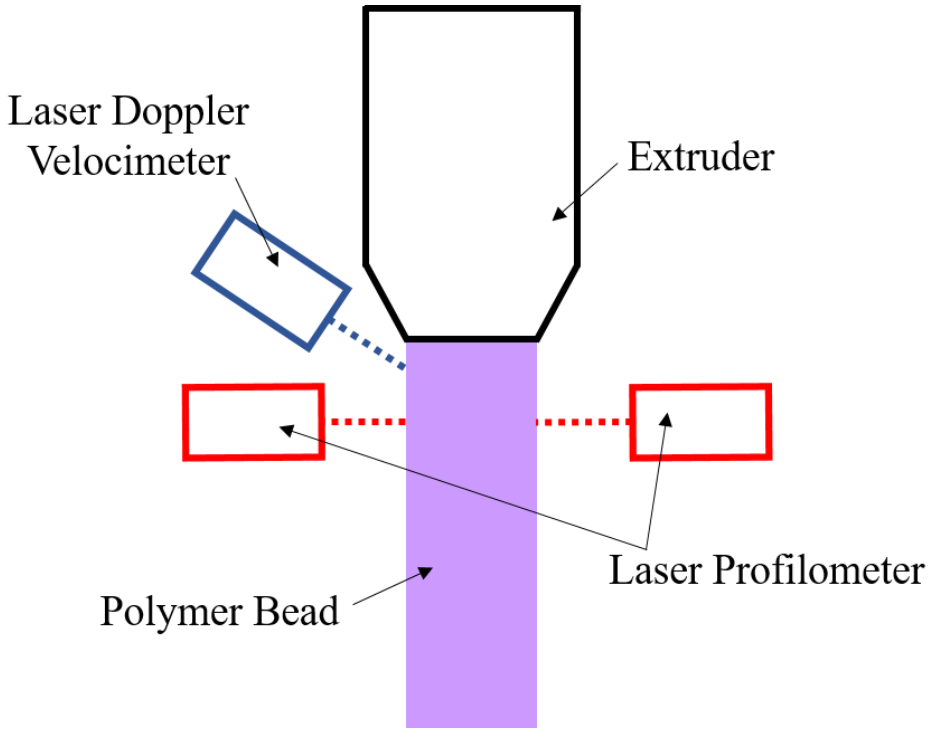


Figure 40. Alternative Setup for Measuring Flow Rate

5.4.1 Drawbacks of the Proposed Method

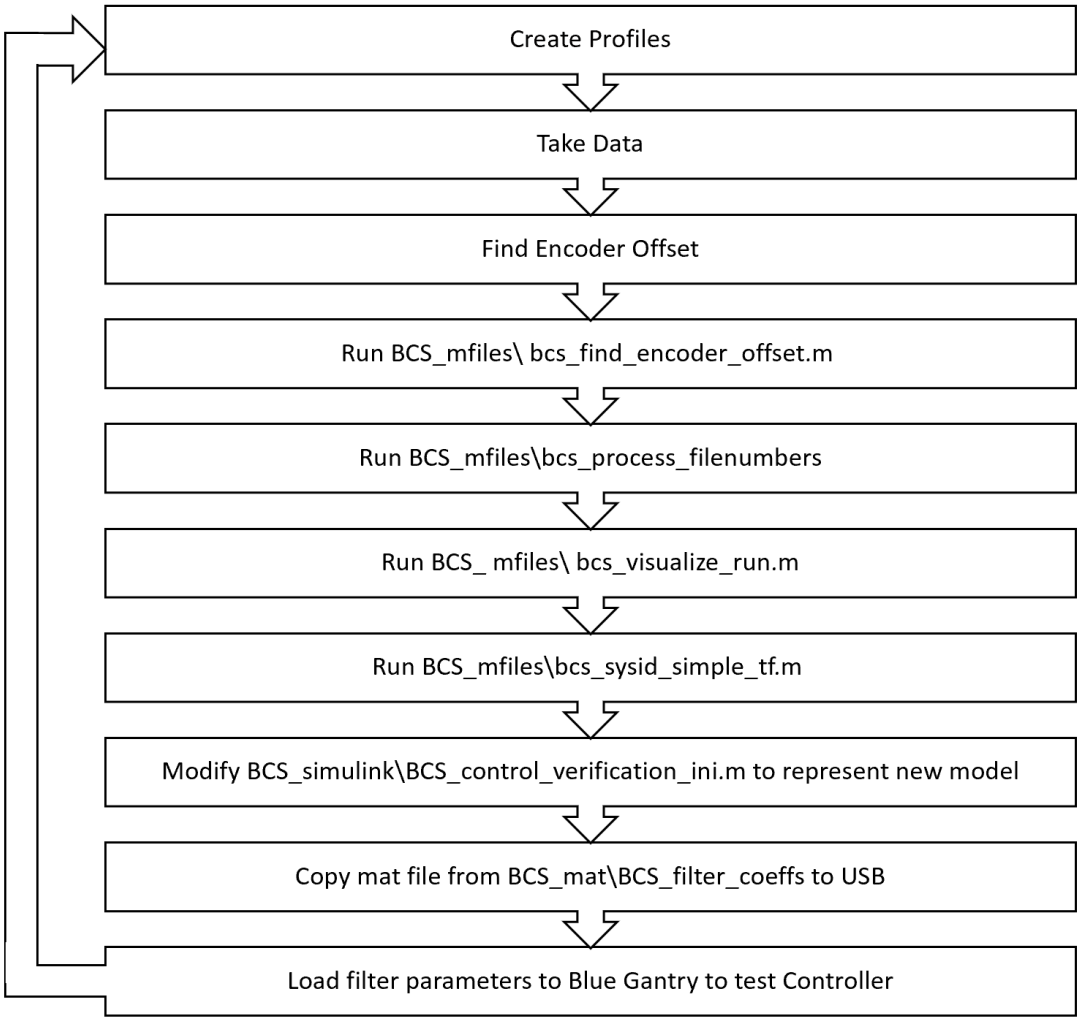
While the proposed setup may be better at identifying the starting and stopping conditions of the extruder dynamics, this setup is not without flaws. The two measurements, one from the LDV and one from the laser profilometer would need to take place after the swell out of the nozzle. If this setup is used, the polymer will stretch under its own weight. Therefore, the measurements will need to take place very close to the point of extrusion. Additionally, this setup does not enable measurement of the dynamics at play during the printing process, which are important to know when designing a controller to be implemented in the future. Furthermore, the group at ORNL has tried to use a LDV to measure the bead velocity in the past and had trouble obtaining reliable measurements from the LDV due to smoke emitted from the extruder during the extrusion process. It may be possible to find a different

LDV that is more suitable for the environmental conditions that are present in the printing process.

APPENDIX A. DOCUMENTATION FOR OAK RIDGE TO CONTINUE WORKING ON BCS

Data Collection and Processing Overview

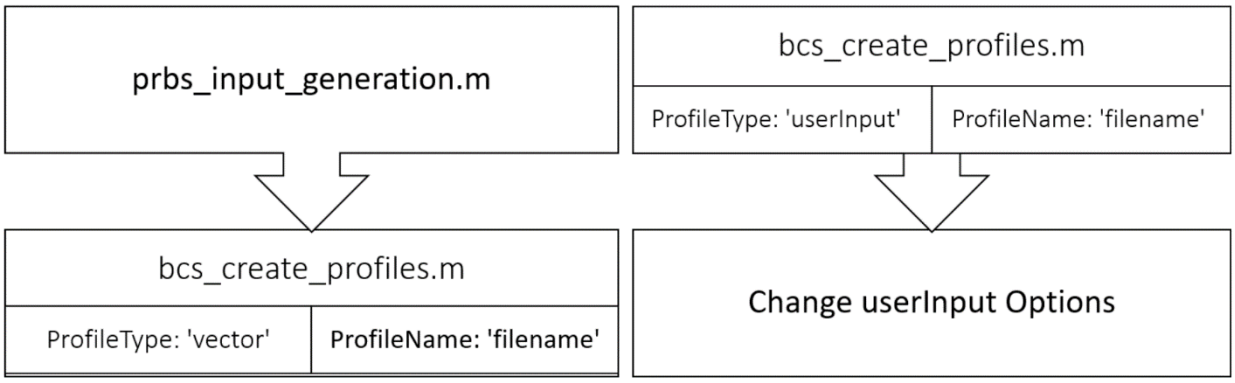
- **All the file paths start from within the Matlab Code Folder**



- When taking data, there is an excel spreadsheet in MatlabCode/BCS_data/Profile Notes.xlsx which has information about each of the runs. Keep track of notes there

Profile Generation for Blue Gantry

Process Flow



BCS_mfiles/ProfileGeneration/bcs_create_profiles.m

User Input:

- profileType:
  - 'vector': Uses variables that were created by another script or are in the workspace to create the profile
  - 'userInput': will use the rest of the User Input Section to create the run
    - This will create a pattern similar to that created in the GUI where you specify an average voltage and deviations from that voltage
- profilename: name for the mat file which is saved to BCS_mat/BCS_profiles/[profilename].mat
  - Change this for each new profile that you want to save
- Changing the wheel profile
  - holdTime: time you want the wheel not to rotate at the end of a run
  - whlCmdHold: commanded wheel voltage when turning the extruder on or off
  - endRotTime: extra rotation time at the end of the path
  - beginningNoRotTime: time in seconds at the beginning of the extrusion where the wheel does not rotate, helpful for allowing the material to stick to the wheel
- Values that matter when using 'userInput' profileType
  - repetitions: (minimum 1) How many times the pattern executes
  - tBtwn: time between repetitions of the pattern (seconds)
  - umid: Midpoint voltage that deviations will be centered around (Volts)
  - tReturnToMid: Time in seconds between upward and downward deviation

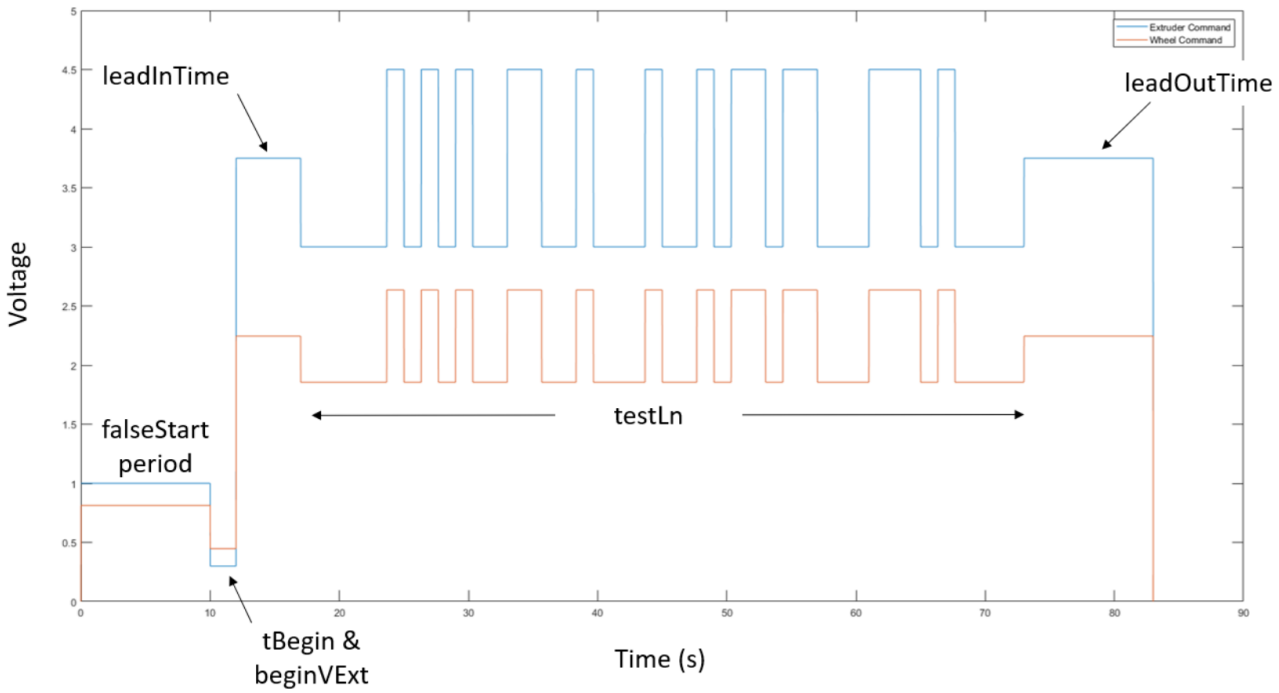
    - tEntry: Time in seconds spent at umid in the beginning
    - tExit: Time in seconds spent at umid at the end
    - dev: 1xn vector of the deviations from umid (Volts)
    - timeDev: 1xn vector of the time spent in each deviation. In other words time spent in the upward portion of the deviation, the length of the pulse up and down will be (2*timeDev + tReturnToMid)
- Wheel Voltage Parameters (provides values for interpolation of the wheel voltages)
    - minWhlV: minimum voltage for the wheel (not really the minimum, just provides the values for interpolation)
    - maxWhlV: maximum volage for the wheel (not really the maximum, just provides the values for interpolation)
    - minExtV: minimum voltage for the extruder (not really the minimum, just provides the values for interpolation)
    - maxExtV: maximum voltage for the extruder (not really the maximum, just provides the values for interpolation)
    - multFactor: with the interpolation points above what do you want to multiply the output wheel speed by. Helpful when you are in the ballpark and just want to make small adjustments to the wheel speed

Outputs:

- A graph so you can see what the profile will look like and decide if you want to keep it
- A .mat file located in “BCS_mat/BCS_profiles/[profilename].mat” which can be loaded into the SIgui using the “Load mat Files” portion of the GUI

BCS_mfiles/ProfileGeneration/bcs_input_generation.m

Function Overview:



- Some of the inputs can be a little confusing the main parameters to change are bold, and a figure of the different regions can be seen above
- After you run this script run bcs_create_profiles.m

User Input:

- Band: I don't really understand this the default is [0 1] and I leave it at that
- **Range**:
  - Voltage Range of extruder inputs.
  - 1x2 vector that should have the min voltage then the max voltage.
  - When using an rgs, the voltage range may be a little larger than the one specified
- **prbsLen**: samples used in the prbs
- **testLn**: time in seconds that the test will last

- leadInTime: time spent extruding before entering the prbs/rgs sequence
- leadInTime: time spent extruding after exiting the prbs/rgs sequence
- The purpose of the false start period is to try to get a good bead flow then to stop the wheel and start again to try to capture the start up. Usually all of these are set to zero, but you can experiment with them.
    - falseStartTime: time in seconds
    - FalseStartWhlV: wheel voltage during the false start
    - FalseStartExtV: extruder voltage during the false start
- tBegin: delay to start the wheel rotating before the extrusion occurs
- beginVExt: voltage given to extruder during the tBegin time
- **type**:
    - 'prbs': will create a pseudo random binary sequence profile
    - 'rgs': will create a random gaussian signal profile

Outputs:

- This function will write multiple values into the workspace of matlab.
- You do not need to do anything with those. bcs_create_profiles.m will use uVec and tVec to make the profile to be saved to a .mat file for running on the Blue Gantry

Procedures:

Wheel Setup

- Turn on heater
- Spin wheel
- When the wheel temperature indicates about 1.7-2.2 the material should stick pretty well to the wheel

Extruder Offset Procedure:

- Start with the wheel in a stopped position with no material on the wheel
- Click the "Motor On" Button and watch the wheel
- Align the extruder in the place where material will be extruded, then lift it in Z slightly.
- Turn on the laser profilometer
- With the wheel stopped, place the pin under the center of the extruder
- Click the "New Test" button in the LabVIEW Front Panel

- Make sure that both squares are set to cRIO control
- Click the “Take Data” button in the LabVIEW Front Panel and make sure the indicator for “Recording Data” is bright green.
- Type 1 in the Whl Cmd box next to the “Motor On” button and press [Enter]
- Once you see a peak in the Width/Height Graph on the front panel type 0 into the motor control box and press [Enter] to stop the wheel
- Click the “Take Data” button in the LabVIEW Front Panel to stop taking data. The indicator for “Recording Data” should turn dark green
- Repeat this procedure about 6 times to ensure you are getting good measurements

Data Processing

BCS_mfiles/bcs_find_encoder_offset.m

Function Overview:

- This Script should be run when the encoder offset needs to be found
- Follow the “Find Encoder Offset” procedure and then run this script
- It is hard to have a range of less than about 600 counts difference between the different runs
- It finds the maximum Area that the laser sees, and reports the number of encoder counts passed at that point in time
- Extruder Offset: The amount of clicks between the place of extrusion and the laser profilometer measurement

User Input:

- filenumber: 1xn vector of the filenumbers to use for finding the encoder offset
- gearRatio: for the gearboxes currently being used this will be either 20 or 60

Function Outputs:

- rangeOfCtsPassed: the range of counts passed between the files. If the range is really large (much greater than 600) or the rangeInInches is too large then you may want to go through the filenumbers one by one and find the outlier
- rangeInInches: converts the rangeOfCtsPassed to inches on the wheel
- avgCtsPassed: this reports the average number of counts passed. Use this number in bcs_process_filenumbers.m

BCS_mfiles/bcs_process_filenumbers.m

Function Overview:

- This function should be the first file run once you take data.
- It takes the text file from the cRIO and creates a '.mat' file which is read by the other functions in MATLAB to do data processing
- The files created will be have the following path "BCS_mat\BCS_[filenumber].mat"

User Input:

- sampleTime: sample time in seconds (1/Hz) that is being used
- gearRatio: for the gearboxes currently being used this will be either 20 or 60
- filenumbers: the filenumbers that you would like to process (Recommended to do just one file when this file has not been run yet in this instance of MATLAB being open to allow bcs_cro_read_data.m to compile. Then enter a 1xn vector of filenumbers.)
- clicksToLaser: clicks between the extruder head and the laser. Get this value from bcs_find_encoder_offset.m. Then enter the value from that script into this field.

Outputs:

- The script will display "Files Processed!" once all of the files have been interpreted. At this point all of the files have been processed and other scripts can be run.

BCS_mfiles/bcs_visualize_run.m

Function Overview:

- This function is typically run after you take data to quickly see what the data looks like without doing any analysis

User Input:

- filenumbers: the filenumbers that you would like to see

- plotCharacteristics: a 1xn vector of tnumbers of the data that you would like to see. There is a list of numbers above this input which are the choices of what you can have plotted
- xLimVals: The times which you want plotted
- viewWheelResponse:
    - true: will display a second figure of the wheel voltage and wheel velocity in RPM
    - false: will not display a second plot

Outputs:

- A plot of the plotCharacteristics chosen

BCS_mfiles/bcs_sysid_simple_tf.m &

BCS_mfiles/bcs_sysid_simple_tf_cincinatti.m

Function Overview:

- The file with "_cincinatti" will not show the extruder voltage in the graphs, otherwise they are the same
- This function will take two filenumbers create a model from one of them and then fit that model to a different data set.
- This function creates a model between the extruder speed and volumetric flow rate out of the nozzle.

User Input:

- filenumberSysID: This is the filenumber that you would like the model to be made for
- filenumberCheck: This is the filenumber that you would like the model to be checked against
- sampleTime: sample time in seconds (1/Hz) that is being used
- timeInterval: The time interval of the data that you would like the model to consider for model creation
- timeIntervalValidateion: The time interval of the data that you would like the model to consider for model validation
- validate:
    - true: if you would like to check the model against the validating file

    - False: if you do not want to validate the model against another file
- saveFigure:
    - True: if you would like to save the figure
        - The figure will be saved to the following path "\BCS_figures\Flow_Rt_Sys_ID\Profile[filenumberSysID]_Cmp_Profile[filenumberCheck].png"
    - False: if you do not want to save the figure
- np: number of poles in the model
- nz: number of zeros in the model
- iodelay:
    - NaN: unknown time delay
    - Any number: a known time delay
- plotsTogether:
    - Set to true if you want a plot with the model creation and validation together
    - Set to false if you want separate plots of model creation and validation

Outputs

- A figure will appear once the script has finished executing. On the left hand side will be the model creation and the right hand side will be the model validation if plotsTogether is true. If plotsTogether is false, then two separate plots will pop up.
- In the Command Window of MATLAB there will be a 'sys' output here it will show a model of the transfer function used along with the Gains, Time Delay, Zeros, and Poles. The model shown will different depending on the fitType used. These values can be used in the Simulink model in order to simulate the flow and controller.
- To get the numerator of the transfer function:
    - Type sys.num into the command window
- To get the denominator of the transfer function type "sys.den" into the command window

BCS_mfiles/bcs_sysid_simple_screw_sys.m

Function Overview:

- This function will take two filenumbers create a model from one of them and then fit that model to a different data set.
- This function creates a model between the extruder commanded voltage and the extruder speed

User Input:

- filenumberSysID: This is the filenumber that you would like the model to be made for
- filenumberCheck: This is the filenumber that you would like the model to be checked against
- sampleTime: sample time in seconds (1/Hz) that is being used
- timeInterval: The time interval of the data that you would like the model to consider
- checkSystem:
    - 'true': will check the model against the filenumberCheck
    - 'false': will not do a verification against filenumberCheck and will not output the second figure
- fitType: parameters for fitting a model to the data.
    - Most likely choice for this fitType will be 'P1D' because a motor is almost always a first order system with a time delay.
    - These parameters are fed into the MATLAB 'procest' function, documentation can be found here under 'type-Process model structure'.

Outputs

- Two figures will appear once the script has finished executing.
    - Figure 1: Model Creation
        - This will show the input voltage and model of the extruder speed with the actual motor speed as well as the model for the motor speed.
    - Figure2: Model Validation
        - This will show the input voltage from the filenumberCheck file with the actual and simulated response.
- In the Command Window of MATLAB there will be a 'sys' output here it will show a model of the transfer function used along with the Gains, Time Delay, Zeros, and Poles. The model shown will different depending on the fitType used. These values can be used in the Simulink model in order to simulate the flow and controller.

BCS_simulink/BCS_control_verification_ini.m

Function Overview:

- This function is used to compile the mat file to be loaded into the blue gantry to make a controller

- As gain scheduling is desired, it will be necessary to modify this file as well as the blue gantry code
- This function is also run before the simulink model which is probably not necessary to run for most applications

Explanation and Inputs for Each Section of Code:

First Section:

- outputMatFile:
    - true: will output a mat file to BCS_mat/[filename]
    - false: will not output a mat file
- filename: a string of the filename you want the mat file to have
- Ts: the sample time for simulink

Load Inputs:

- Explanation:
    - Use this if you want to run the simulink portion of the code or else it doesn't matter
- filenumber: the filenumber of a run that has already been run that you want to use for the extruder commands

Wheel Parameters:

- Explanation:
    - Only important if you want to do a simulation in Simulink
    - Creates the transfer function for the wheel
    - Get these values from the wheel system identification
- Kw: Gain for the wheel transfer function
- tau_w: the time constant for the wheel

Transfer function G1 from Extruder Voltage to Motor Speed:

- Explanation:
    - Creates the parameters for the transfer function from the extruder voltage to motor speed
    - Run bcs_sysid_simple_screw_sys.m to get these values. They will be output to the command window.

- K1: Kp from command window
- tau1_p1: Tp1 from command window
- tau1_p2: Tp2 from command window
- timeDG1: Td from command window
- tau1_z1: Tz from command window

Transfer Function G2 from Extruder Motor Speed to Flow Rate:

- Explanation:
    - Creates parameters for transfer function from screw speed to flow rate
    - Run bcs_sysid_simple_tf.m to get these parameters
- beta: run bcs_sysid_simple_tf.m and type ‘sys.num’ into the command window. The vector goes into beta
- alpha: run bcs_sysid_simple_tf.m and type ‘sys.den’ into the command window. The vector goes into alpha

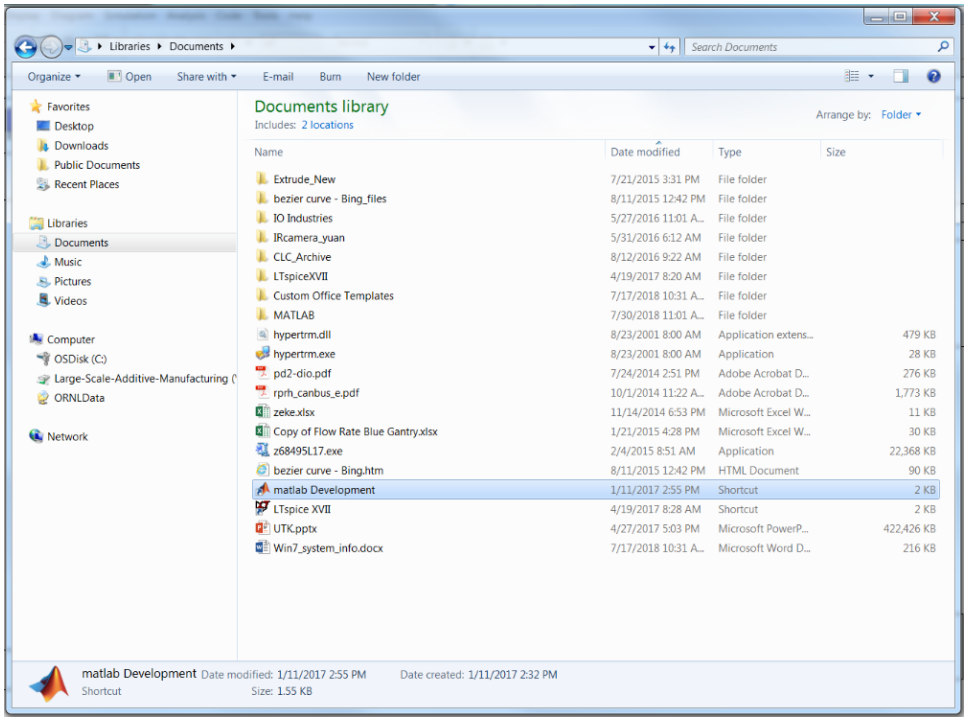
Desired Poles:

- Use this to change the wheel response to put the poles where you want them. Change the values of tau_d1 and taud_2 to place the poles in the desired locations. Then run the Simulink code to see if those values work well. 0 and 0.01 work well to control the wheel

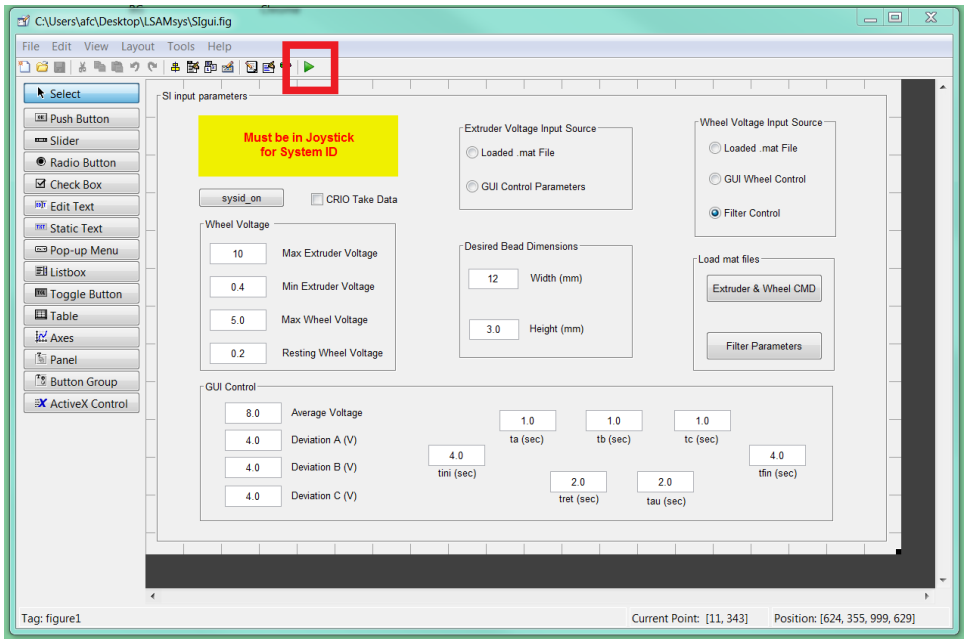
Blue Gantry GUI Operation

Startup on the Blue Gantry:

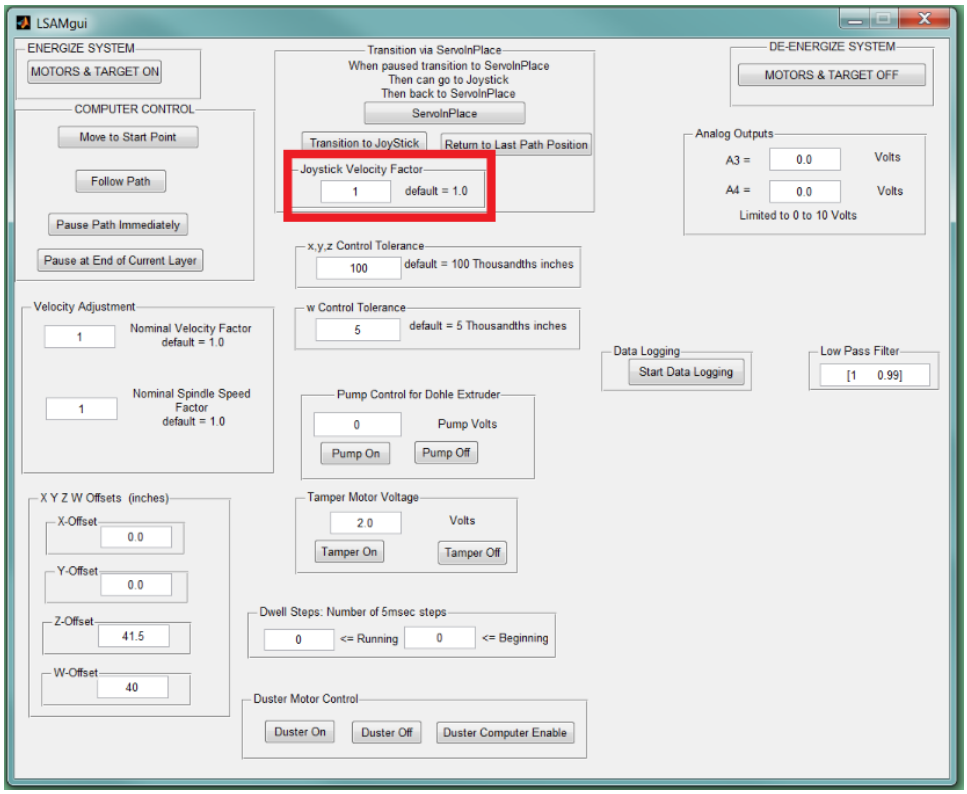
- Launch MATLAB by going to the Documents Folder and Pressing “matlab Development”



- Launch the SI GUI by pressing the play button in the window



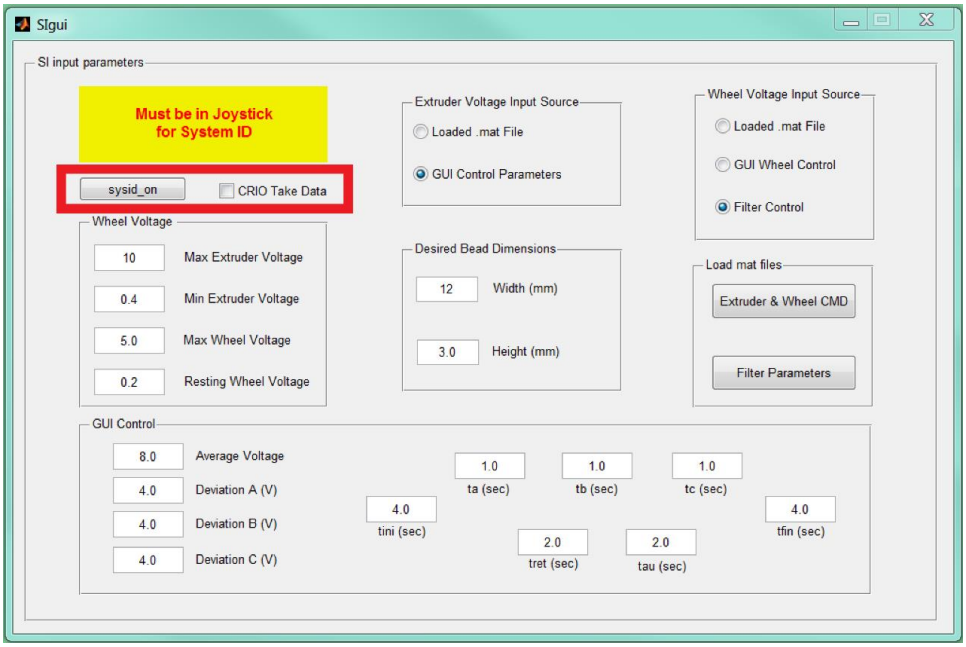
- Build the main LSAM GUI as usual to connect to the target.
- Use the Joystick to move the extruder to the appropriate location on the build platform. Change the Joystick Velocity Factor to slow the joystick down (typically use 0.1 for fine movements)
    - MUST PRESS ENTER AFTER TYPING NUMBER INTO THE BOX



- Use the readouts on the target screen to make sure you are in the correct measured (not commanded) X, Y, Z, W location
- IMPORTANT: Press enter in all the SIgui boxes, and make sure to toggle each of the radio buttons at start up to ensure that you know what the value of each portion of the GUI is
    - After any action is performed in the GUI, text verifying the action will appear in the command window

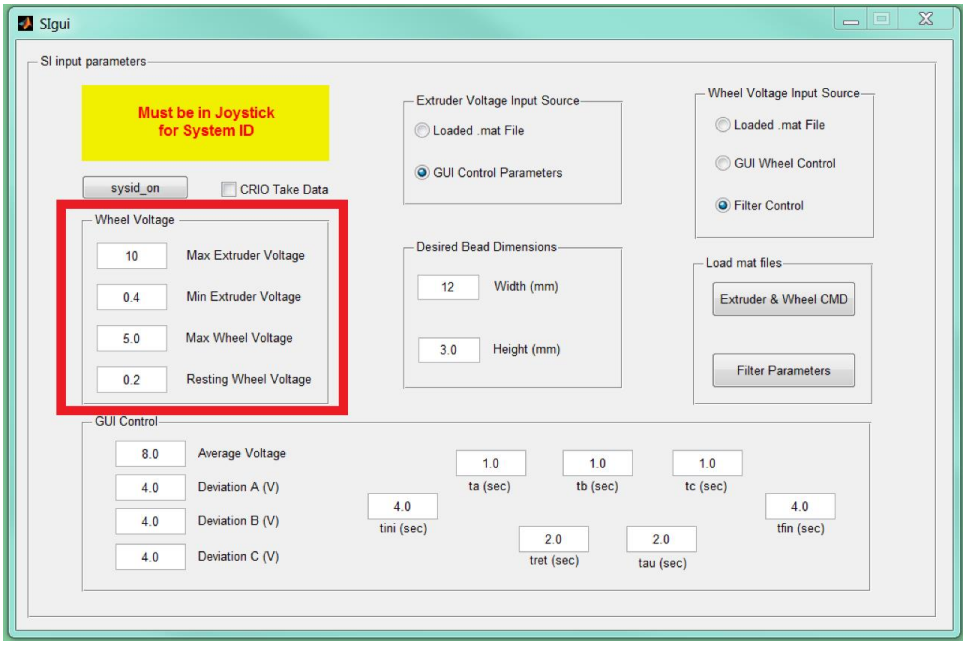
Sections of GUI:

Data Collection



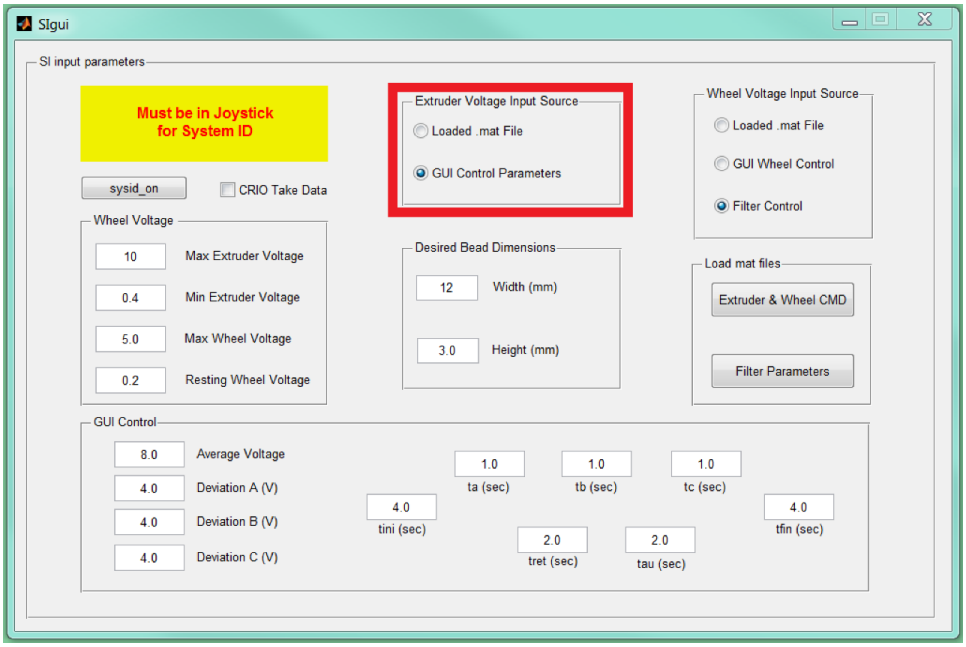
- sysid_on button
    - Used to start a test run. Pressing the button once will start a test.
    - After the test is complete, the button must be pressed again to reset the button
- CRIO Take Data Checkbox
    - Checked: Will initiate the trigger in LabVIEW if set to “BG Master” Mode to take data when a test is being run (sysid_on button will activate the trigger once pressed)
        - Remember to press the “New Test” button in LabVIEW between tests
    - Unchecked: Will not activate the trigger in LabVIEW to take data

Wheel Voltage



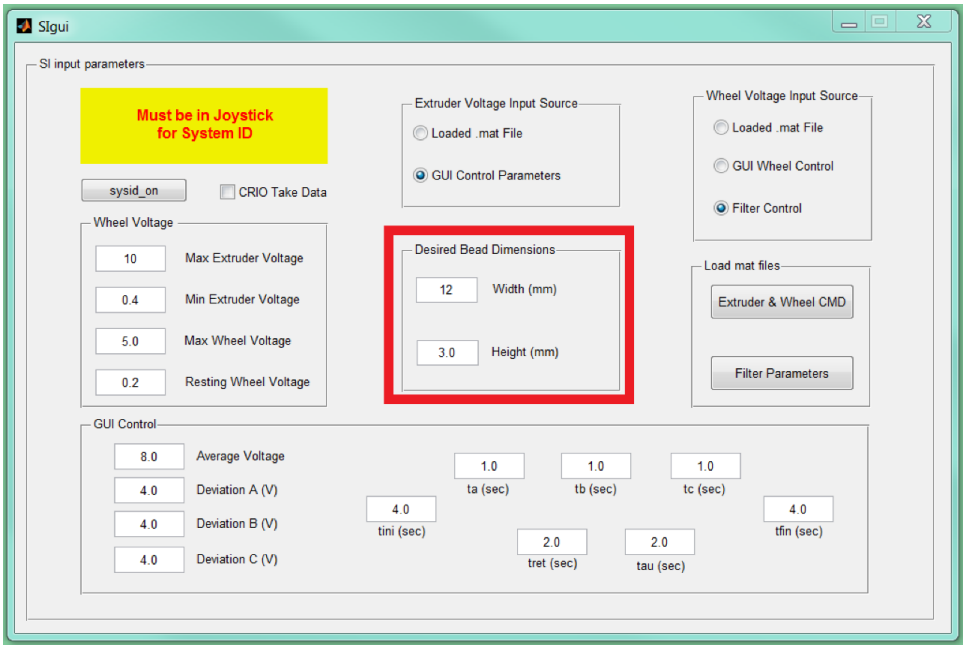
- This section controls the wheel voltage when LabVEIW is in “BG Master” Mode
- If the SIgui is in GUI wheel Control it will do interpolation between the values given for the extruder and wheel voltage values. It interprets the “Resting Wheel Voltage” as the Min Wheel Voltage for the interpolation
    - This is a bit misleading, its not actually the min and max voltage that can be sent, it is the values which are interpolated between, and it will never send more than 10V to the wheel
- Run with Constant Wheel Speed: Set the “Max Wheel Voltage” and the “Resting Wheel Voltage” to the same voltage value
- Resting Wheel Voltage: The voltage that will be sent to the wheel when not in the middle of a run

Extruder Voltage Input Source



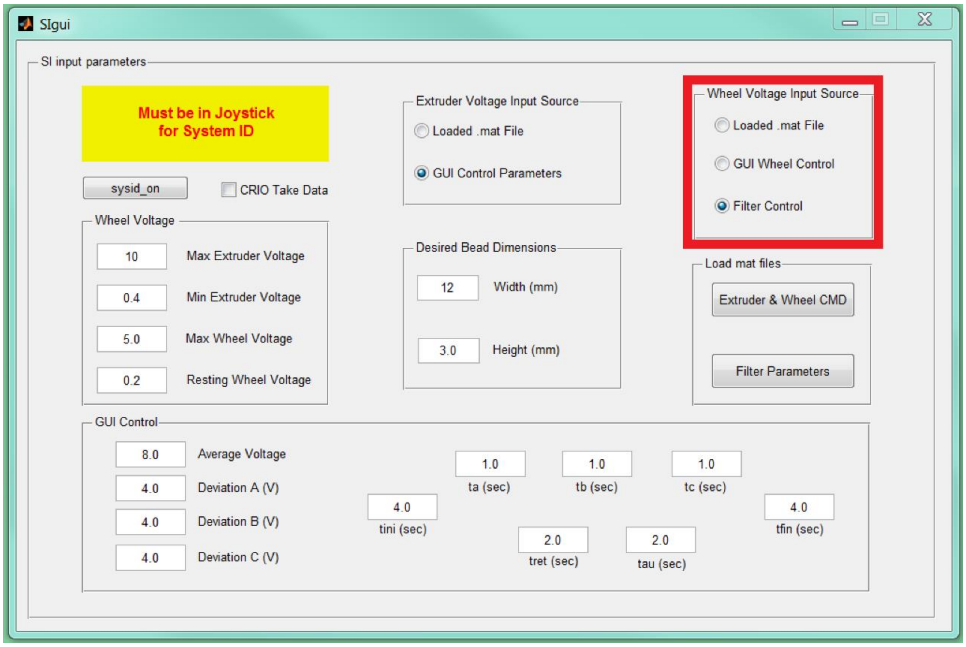
- Loaded .mat File: will use the loaded mat file as the voltage source
- GUI Control Parameters: Uses the GUI Control box as the voltage source for the extruder

Desired Bead Dimensions



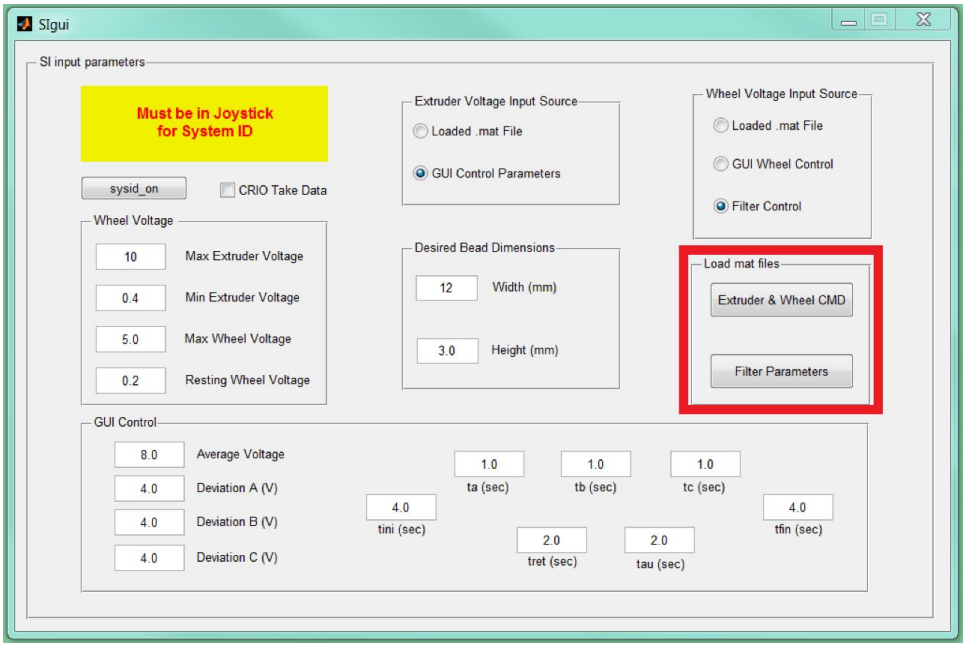
- This only matters when using the Filter Control as the Wheel Voltage Input Source
- The scaling may be off given your filter coefficients so experiment to see what gives you a reasonable bead.
    - If the bead width seems to be changing a lot, then you probably have bad filter coefficients, and need to modify your filter

Wheel Voltage Input Source



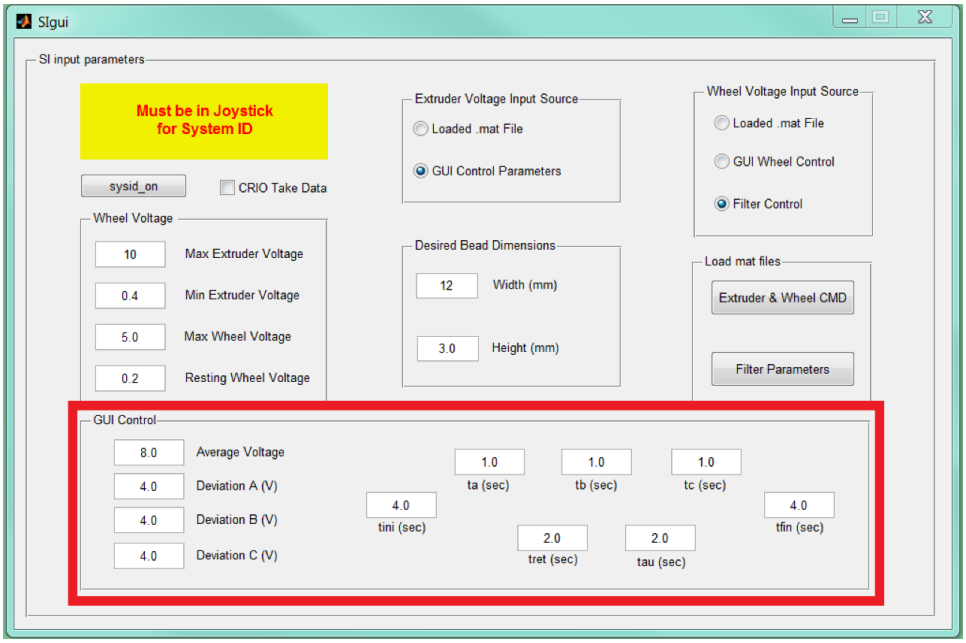
- Loaded .mat file: uses the scaling decided in the .mat file when the profile was made as the wheel voltage source
- GUI Wheel Control: uses the Wheel Voltage inputs for interpolation of the wheel speed
- Filter Control: Uses the filter coefficients and the Desired Bead Dimensions to vary the wheel voltage output

Load mat files

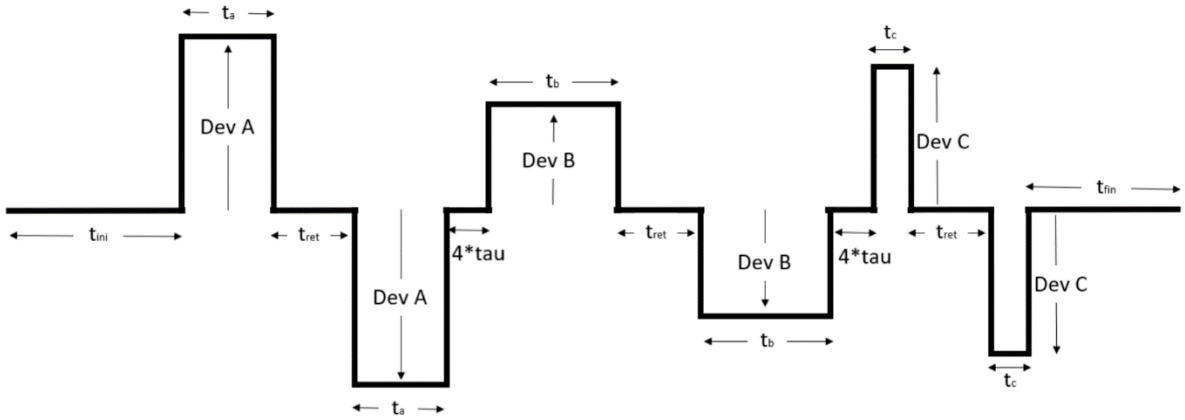


- To use these commands first load the mat file into the command window.
    - You can do this by dragging the file from the windows explorer to the command window
    - After you do this a load command will show up in the command window
    - Then use the buttons in the GUI
- Extruder & Wheel CMD: loads the extruder and wheel commands from a mat file (generated from “bcs_create_profiles.m” to the target
- Filter Parameters: loads the filter coefficients mat file generated from “BCS_control_verification_ini.m” to the target

GUI Control



- Creates a profile like the one shown below, where all the deviations are from the average voltage

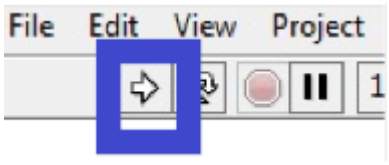


LabVIEW GUI Operation

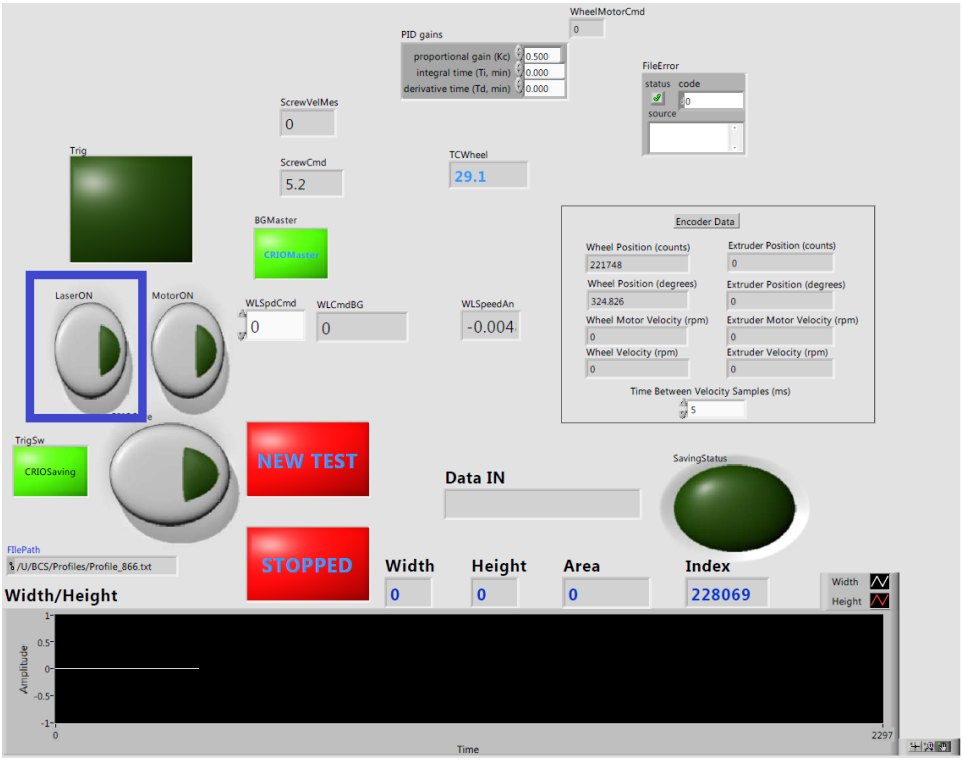
Starting Procedures

- Open LabVIEW

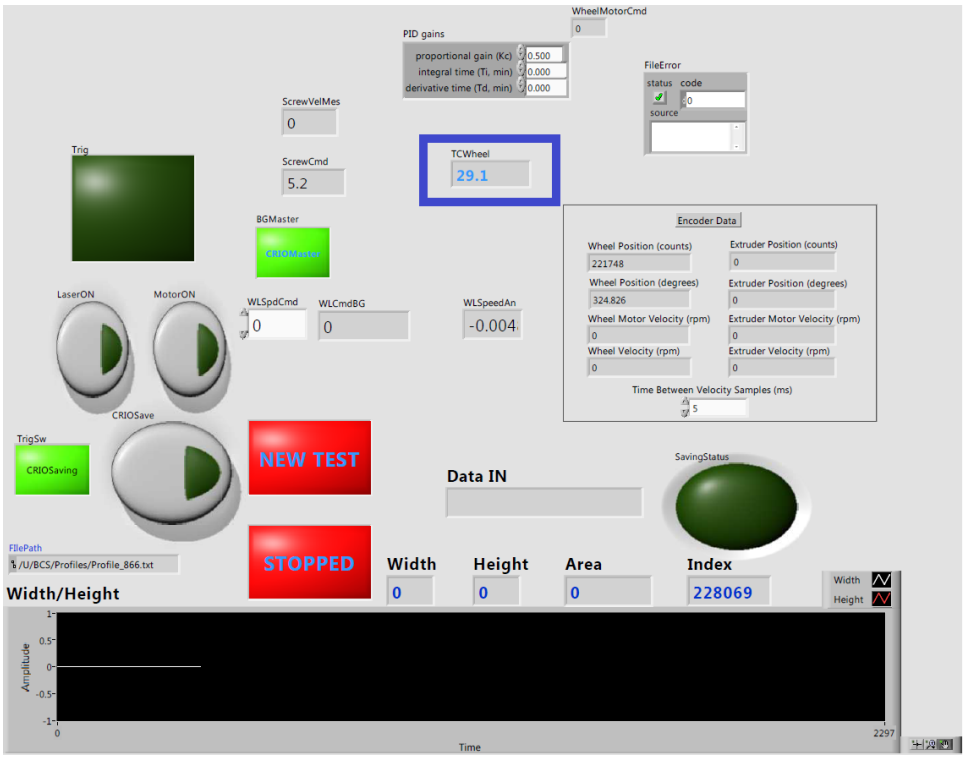
- Navigate to the project to open it
- Open BeadChar4.lvproj
- In the project explorer, press the + next to “NI-cRIO-9067-030d019e (132.168.0.3)”
- Double click “BeadCharSysETHMultiTh with FPGS_Vis.vi” to open the VI
- Press the Run button below the menu bar



- If an error appears clear the error and try to relaunch the program by hitting the run button.
    - Sometimes it will not reconnect until you close LabVIEW and then re-open the program
    - You can also try restarting the cRIO by cycling the power
- Turn the Laser On
    - Press the LaserON button (when the laser is on it will turn bright green)

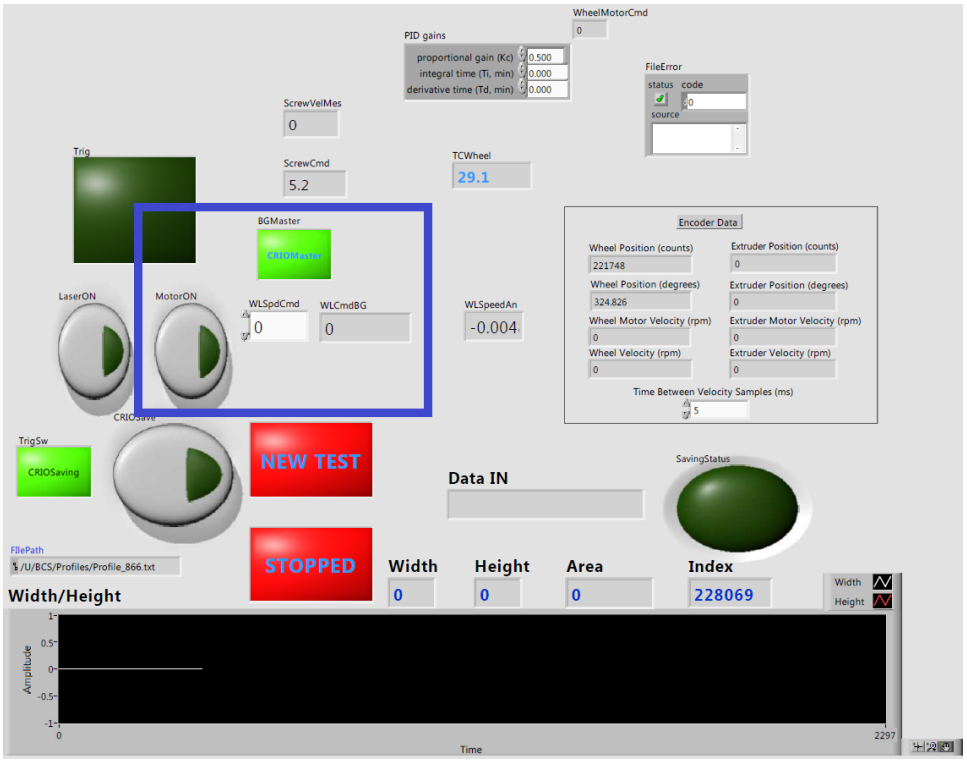


- Turn the Motor On
- Wheel Temperature



- The wheel should be good to extrude onto when the TCWheel is between 1.7E2 and 2.2E2

Motor Operation



Control the Motor through LabVIEW:

- The BGMaster Switch should be bright green and say “CRIOMaster”
- Press the MotorON button
    - The motor can receive power when this button is bright green
- Type a voltage which you want to send to the wheel to in the WLSpdCmd box

Control the Motor through the Blue Gantry:

- The BGMaster Switch should be dark green and say “BGMaster”
- WLCmdBG box will show the value that is being sent

Saving Procedures

<u>Saving from the Blue Gantry:</u>

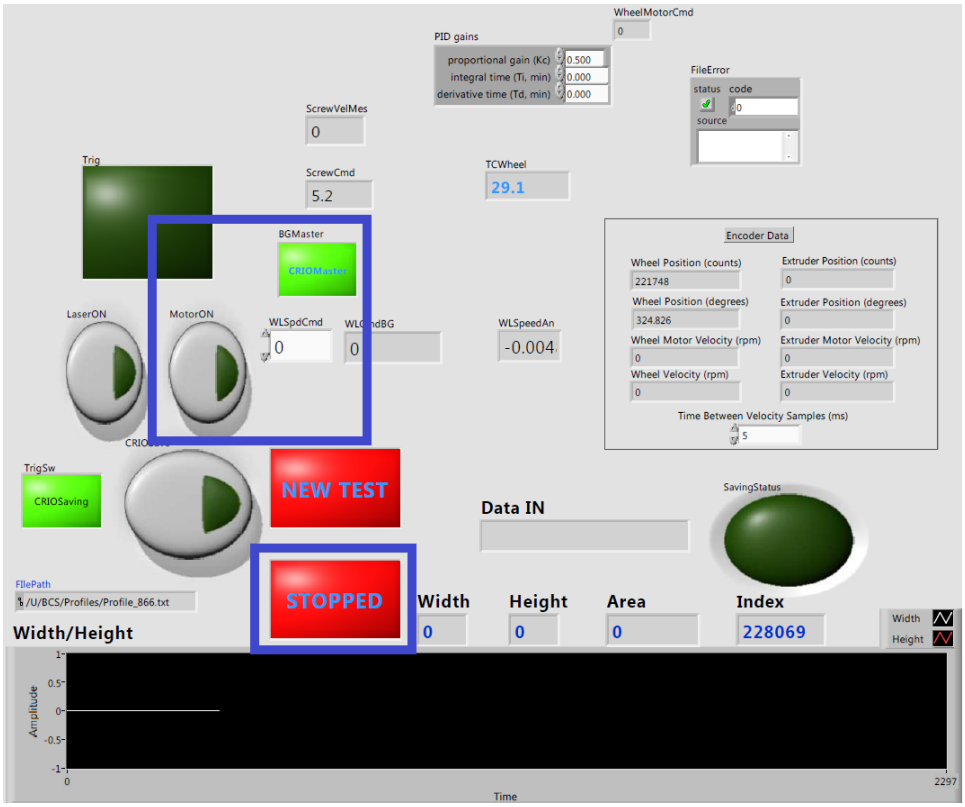
- Have the Trig Sw set to dark green for BG Saving
- Press New Test before each run and pressing the sysid_on button in the Simulink SIgui
    - Make sure that the FilePath increments upwards when you press the button to ensure that it is a new file
- When the cRIO Take Data checkbox is checked in theSimulink SIgui the following will occur during the run:
    - The Trig box should turn bright green
    - The SavingStatus indicator on the right should turn bright green
- After the run, the Trig and SavingStatus should turn dark green

<u>Saving from the LabVIEW Front Pannel:</u>

- Press New Test before each experiment

    - Make sure that the FilePath increments upwards when you press the button to ensure that it is a new file
- Press CRIO Save so the button turns bright green
- The saving status should turn bright green
- To stop saving, press CRO Save again so the button turns dark green.
    - The Saving Status should turn dark green

Closing the LabVIEW



- Put the BGMaster in CRIO Master Mode (bright green)
- Type 0 in the WLSpdCmd box
- Turn the MotorOn Button off (should show dark green state)
- Press Stopped Button
- Make sure the wheel has stopped
    - If it hasn't you will need to restart the program and try to get the wheel to stop in the BGMaster mode
- Close LabVIEW

REFERENCES

[1] M. Mellody and N. R. C. (U.S.), "Novel Processes for Advanced Manufacturing: Summary of a Workshop," 2013.

[2] X. Yan and P. Gu, "A review of rapid prototyping technologies and systems," *CAD Comput. Aided Des.*, vol. 28, no. 4, pp. 307–318, 1996.

[3] C. Holshouser *et al.*, "Out of Bounds Additive Manufacturing," *Adv. Mater. Process.*, vol. 171, no. 3, pp. 15–17, 2013.

[4] C. Duty *et al.*, "What Makes a Material Printable? A Viscoelastic Model for Extrusion-Based 3D Printing of Polymers." 2017.

[5] S. S. Babu, L. Love, R. Dehoff, W. Peter, T. R. Watkins, and S. Pannala, "Additive manufacturing of materials: Opportunities and challenges," *MRS Bull.*, vol. 40, no. 12, pp. 1154–1161, 2015.

[6] C. E. Duty *et al.*, "Structure and Mechanical Behavior of Big Area Additive Manufacturing (BAAM) Materials," *Rapid Prototyp. J.*, vol. 23, no. 1, pp. 181–189, 2017.

[7] L. Love, "Utility of Big Area Additive Manufacturing (BAAM) for the Rapid Manufacture of Customized Electric Vehicles," 2014.

[8] A. A. Hassen *et al.*, "The Durability of Large-Scale Additive Manufacturing Composite Molds." 2017.

[9] C. Duty, V. Kunc, B. Lokitz, and R. Springfield, "Evaluation of Additive Manufacturing for High Volume Composite Part Models." 2017.

[10] "Material Development for Tooling Applications Using Big Area Additive Manufacturing (BAAM)." pp. 1–8, 2015.

[11] C. Rauwendaal, P. J. Gramann, B. A. Davis, and T. A. Osswald, *Polymer Extrusion*, 4th ed. Munich: Hanser, 2001.

[12] N. G. McCrum, C. P. Buckley, and C. B. Bucknall, *Principles of Polymer Engineering*, 2nd ed. Oxford Science, 1997.

[13] "THERMWOOD LSAM - Large Scale Additive Manufacturing," 2018. [Online]. Available: http://www.thermwood.com/lsam_home.htm.

[14] "THERMWOOD LSAM 10 12 Small." [Online]. Available: http://www.thermwood.com/images/2017_site/machines/lsam/thermwood_lsam_10_20_small.fw.png. [Accessed: 10-Jan-2018].

[15] "LSAM Head Composite Image." [Online]. Available: http://www.thermwood.com/lsam/images/lsam_print_head_images/lsam_head_composite_image_2.fw.png. [Accessed: 10-Jan-2018].

[16] R. Lind, C. Blue, L. Love, B. Post, and P. Lloyd, "Enhanced Additive Manufacturing with a Reciprocating Platen," 2018.

[17] "Ferrite Recommendations." [Online]. Available: https://www.a-m-c.com/downloads/.

[18] A. Savitzky and M. J. E. Golay, "Smoothing and Differentiation of Data by Simplified Least Squares Procedures," *Anal. Chem.*, vol. 36, no. 8, pp. 1627–1639, 1964.

[19] R. K. Pearson, "Outliers in process modeling and identification," *IEEE Trans. Control Syst. Technol.*, vol. 10, no. 1, pp. 55–63, 2002.

[20] R. K. Pearson, Y. Neuvo, J. Astola, and M. Gabbouj, "The class of generalized hampel filters," *2015 23rd Eur. Signal Process. Conf. EUSIPCO 2015*, pp. 2501–2505, 2015.

[21] D. P. Allen, "A frequency domain Hampel filter for blind rejection of sinusoidal interference from electromyograms," *J. Neurosci. Methods*, vol. 177, no. 2, pp. 303–310, 2009.

[22] MATLAB, "Signal Smoothing." [Online]. Available: https://www.mathworks.com/help/signal/examples/signal-smoothing.html#d120e350.

[23] J. Seo, H. Ma, and T. K. Saha, “On savitzky-golay filtering for online condition monitoring of transformer on-load tap changer,” *IEEE Trans. Power Deliv.*, vol. 33, no. 4, pp. 1689–1698.

[24] R. W. Schafer, “What is a Savitzky-Golay Filter?,” no. July, pp. 111–117, 2011.

[25] David Trumper, “2.14 Analysis and Design of Feedback Control Systems.” Massachusetts Institute of Technology: MIT OpenCourseWare, https://ocw.mit.edu. License: Creative Commons BY-NC-SA., 2014.

[26] F. Haugen, *Exercises to Basic Dynamics and Control*, no. August. 2010.

[27] D. Trumper and S. Dubowsky, “Natural Response,” Spring 2005. Massachusetts Institute of Technology: MIT OpenCourseWare, https://ocw.mit.edu. License: Creative Commons BY-NC-SA., 2005.

[28] P. Fractions, “TRANSFER FUNCTIONS & STABILITY Partial Fractions : Unique Poles,” no. 190, pp. 2–5.

[29] T. P. Nicholas Hadjiconstantinou, Peter So, Sanjay Sarma, “Second-Order Systems.” Massachusetts Institute of Technology: MIT OpenCourseWare, https://ocw.mit.edu. License: Creative Commons BY-NC-SA., 2007.

[30] J. Stewart, “Applications of Second-Order Differential Equations,” *Calculus*, pp. 1–10, 2012.

[31] A. Mattuck, H. Miller, J. Orloff, and J. Lewis., “Under , Over and Critical Damping.” Massachusetts Institute of Technology: MIT OpenCourseWare, https://ocw.mit.edu. License: Creative Commons BY-NC-SA., 2011.

[32] K. J. Åström and P. Eykhoff, “System identification-A survey,” *Automatica*, vol. 7, no. 2, pp. 123–162, 1971.

[33] T. C. Hsia, *System Identification Least-Squares Method*. 1997.

[34] L. Ljung, *System Identification Theory for the USer*, 2nd ed. Prentice Hall PTR, 1999.

[35] K. J. Åström, “Lectures on the Identification Problem The Least Squares Method,” 1968.

[36] MATLAB, “Goodness of Fit.” [Online]. Available: https://www.mathworks.com/help/ident/ref/goodnessoffit.html.

[37] “MAE and RMSE — Which Metric is Better?” [Online]. Available: https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d.

[38] N. R. Schott, “Analysis of Plastics Extruder Dynamics,” The University of Arizona, 1971.

[39] D. Reber, E. Lynn, and E. Freech, “A Mathematical Model fro Predicting Dynamci Behavior of a Plasticating Extruder,” *Polym. Eng. Sci.,* vol. 13, no. 5, pp. 346–356, 1973.

[40] W. L. Krueger, “Experimental Illustrations of Dynamic Extrusion Theory,” *SPE,* vol. 18, no. 10, 1962.

[41] Micro-Epsilon, “scanCONTROL Configuration Tools 4.0.” pp. 1–172, 2008.

[42] LINDY, “13mm Ferrite Core, Black.” [Online]. Available: https://www.lindy.co.uk/cables-adapters-c1/audio-video-c107/13mm-ferrite-core-black-p8085. [Accessed: 19-Oct-2018].

[43] Polytec, “CABLE.” [Online]. Available: https://www.polytec.com/us/velocimetry/areas-of-applications/cable/.